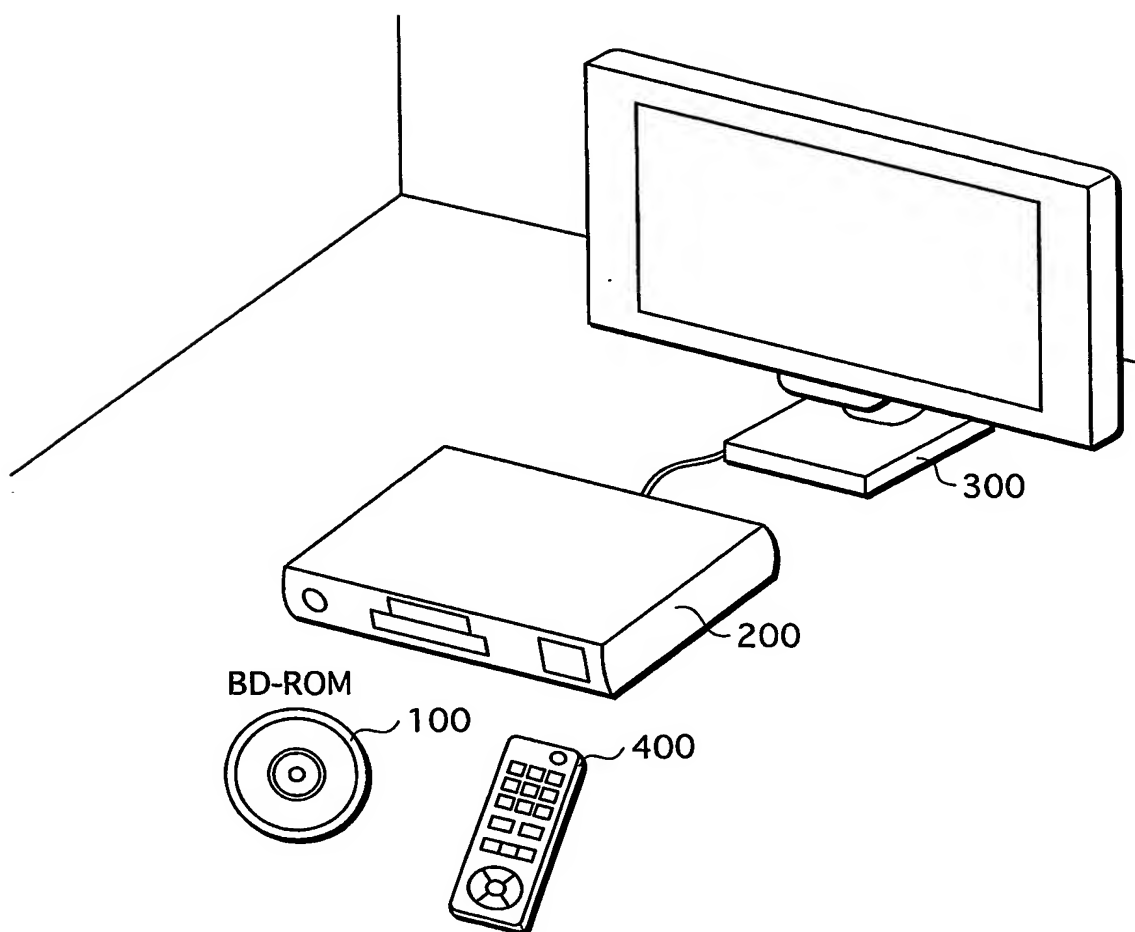
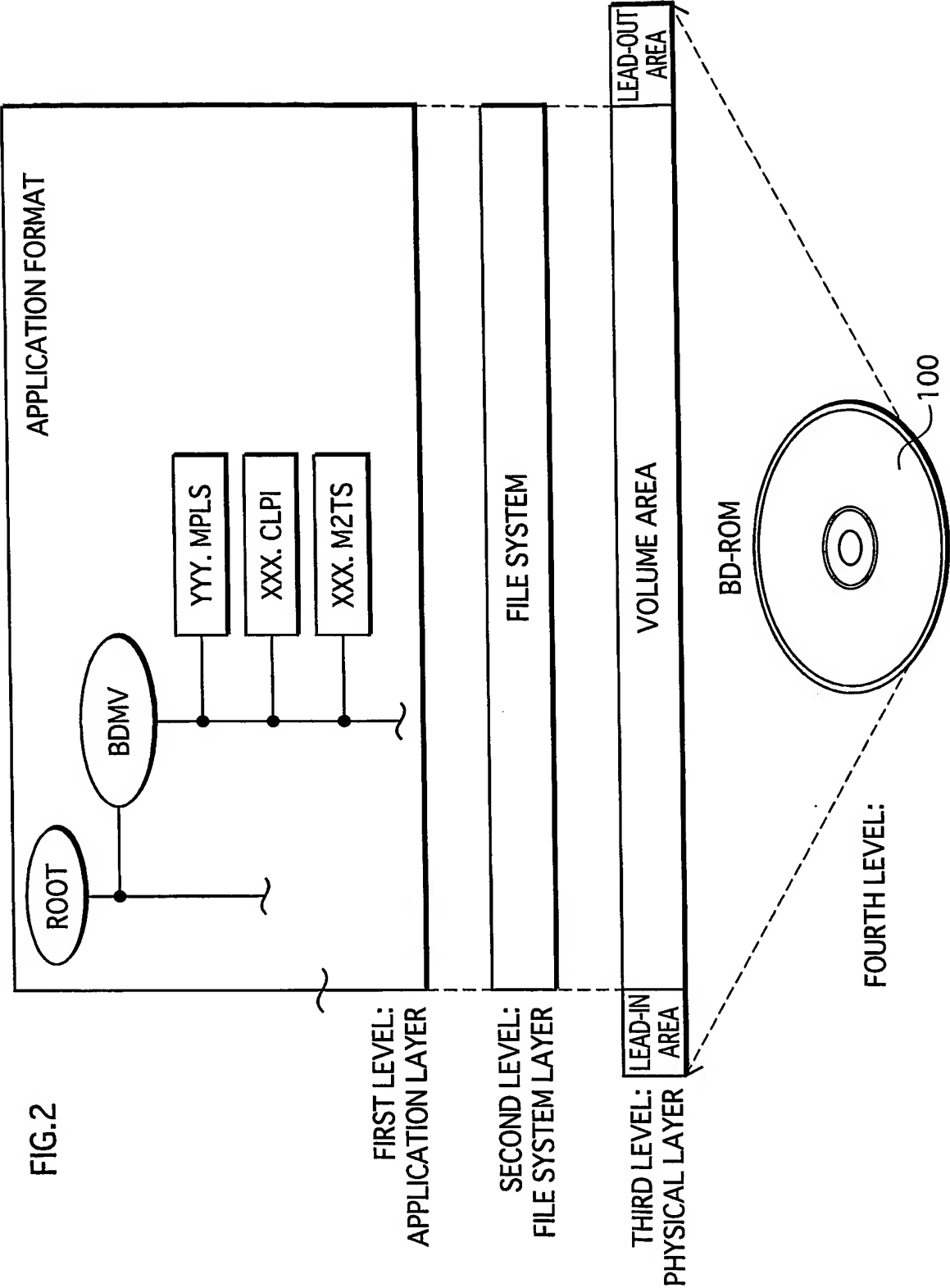
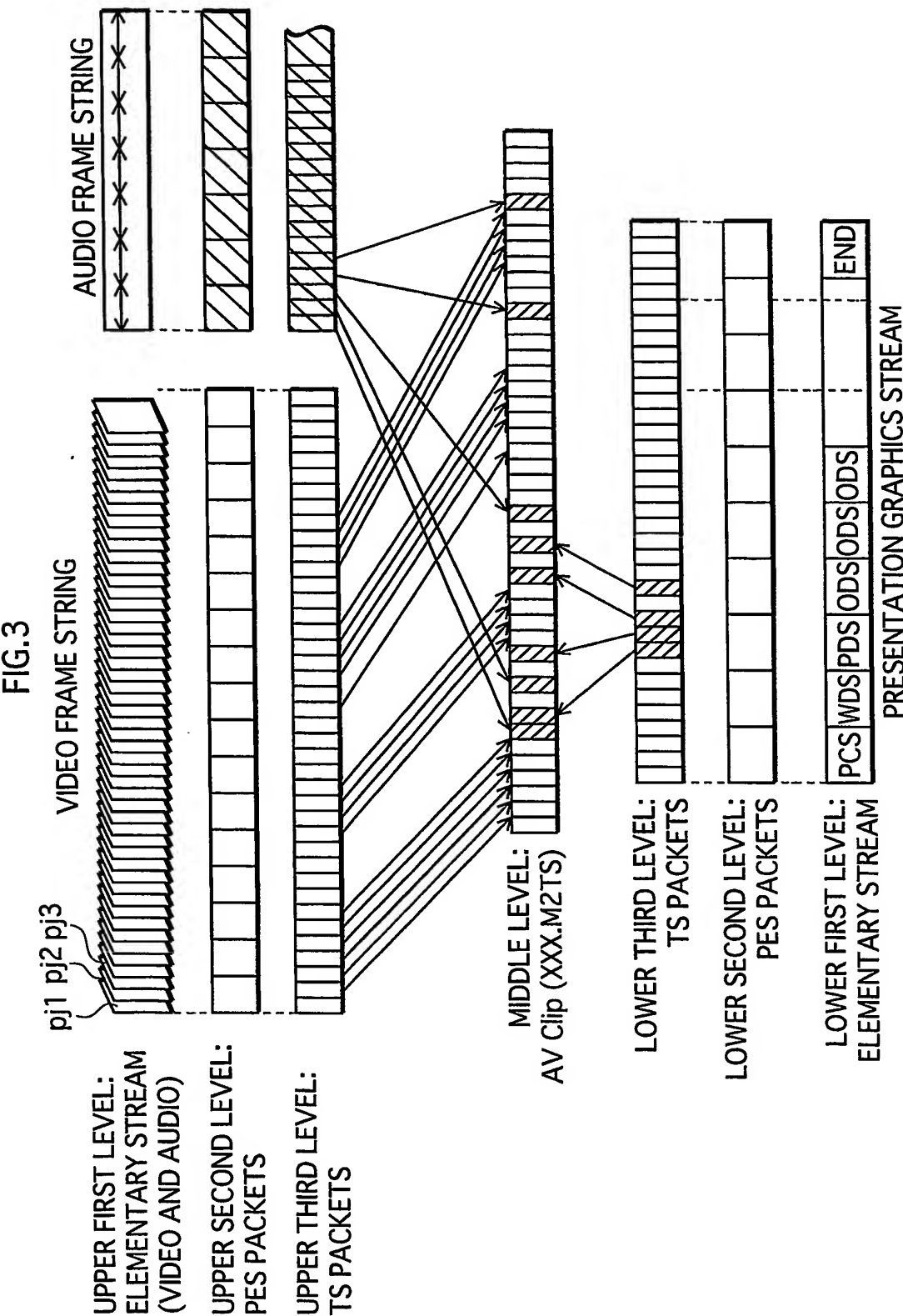
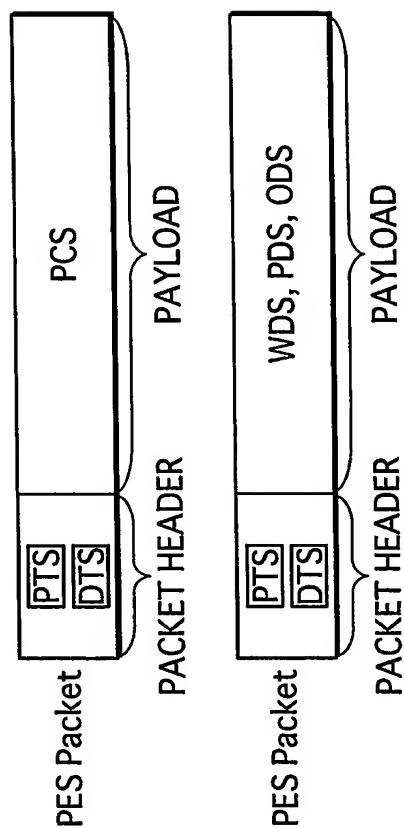
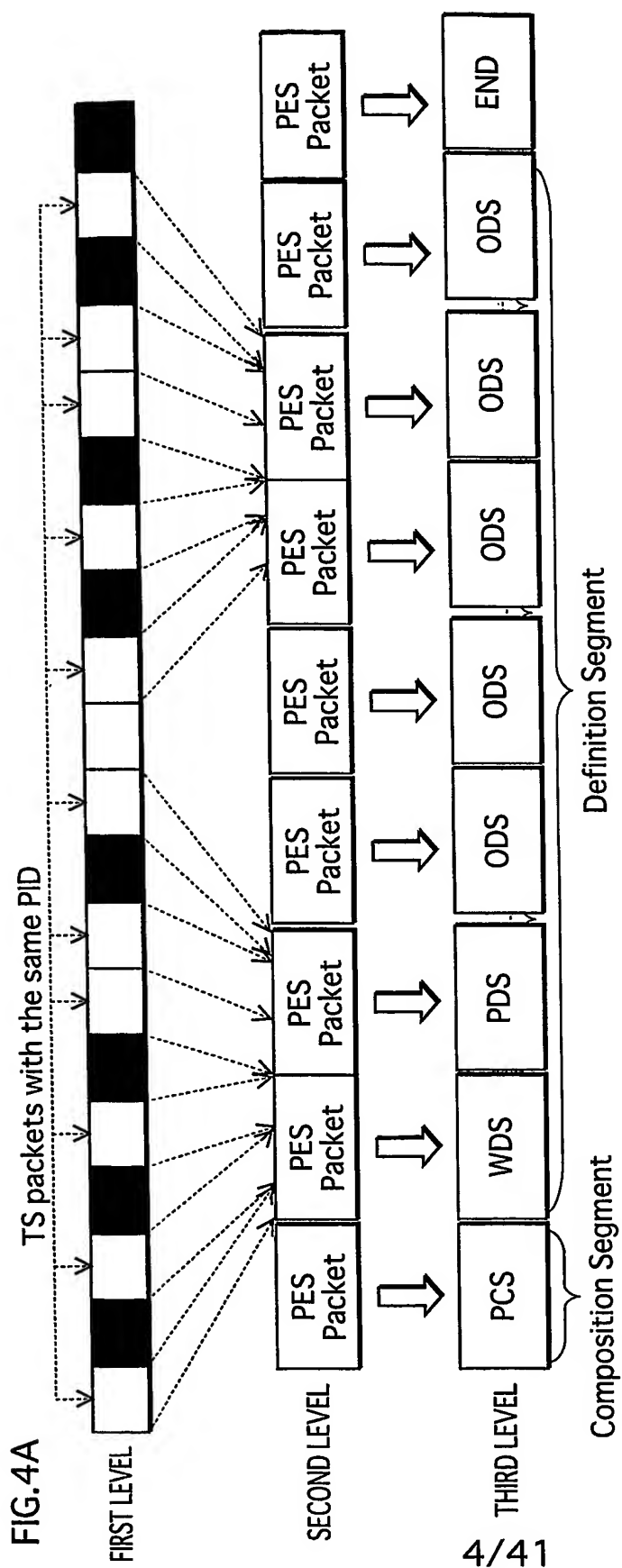


FIG.1



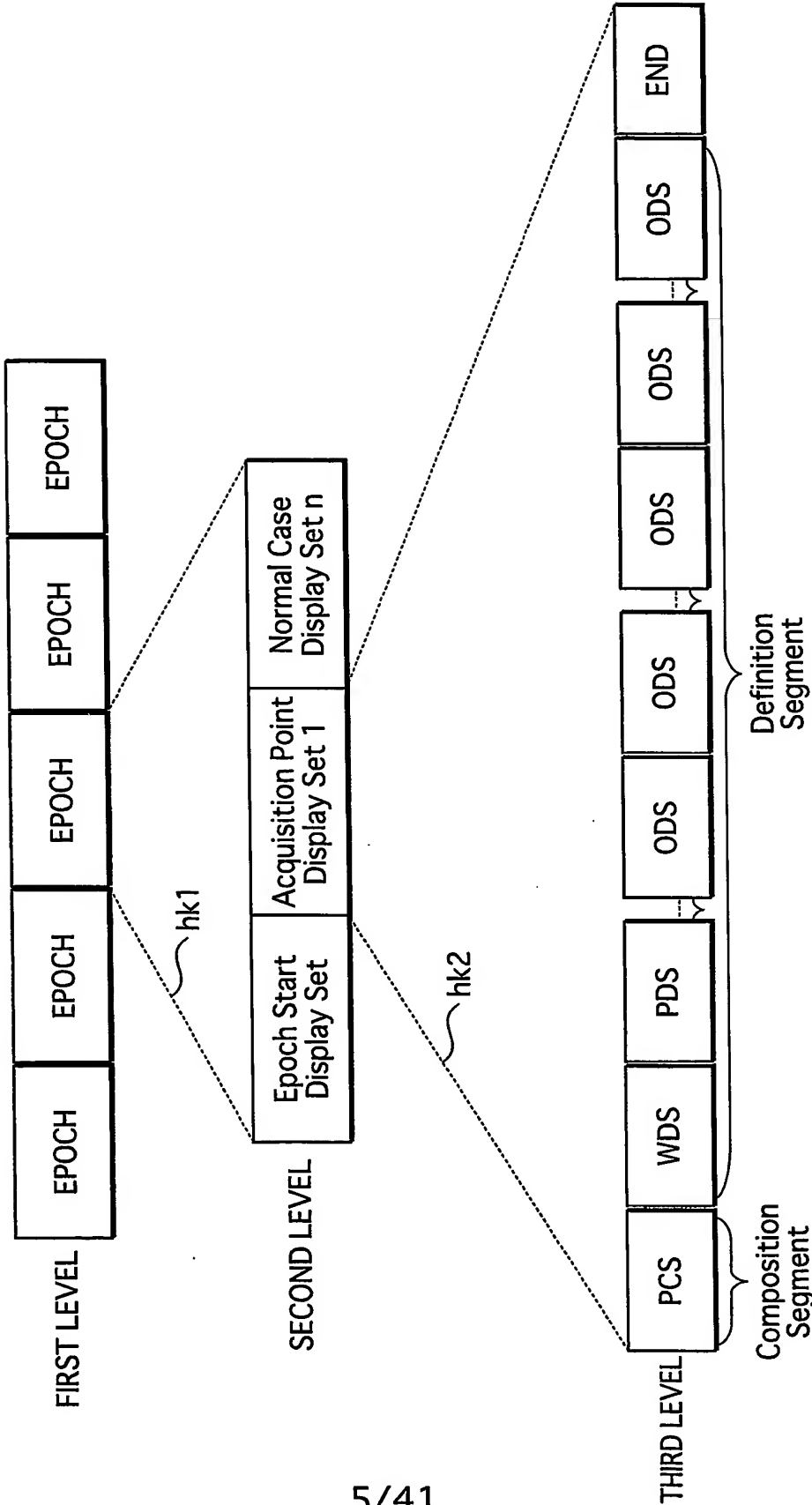






**FIG. 4B**

FIG.5



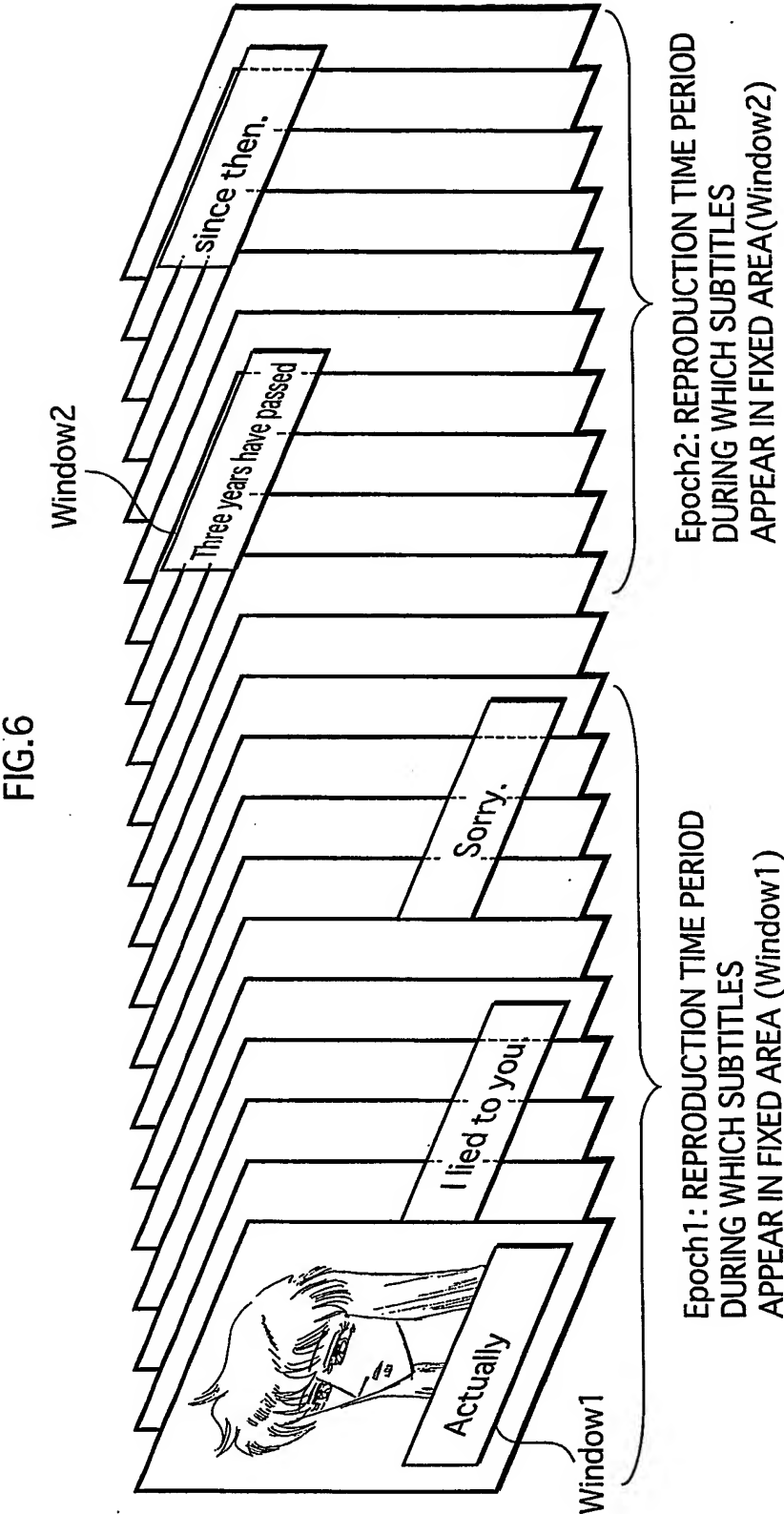


FIG.7A

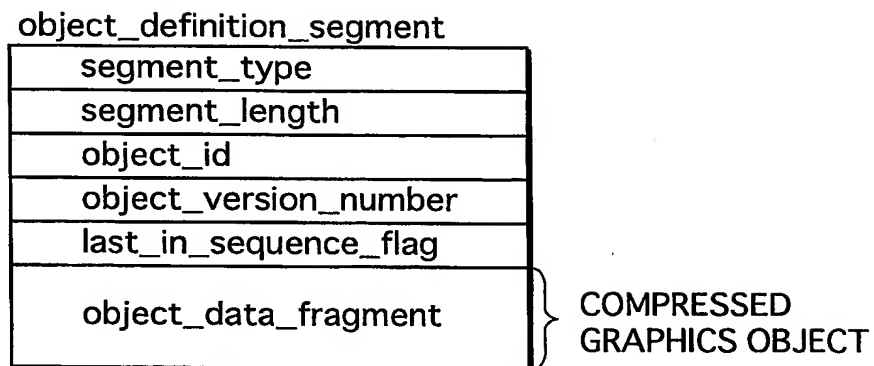
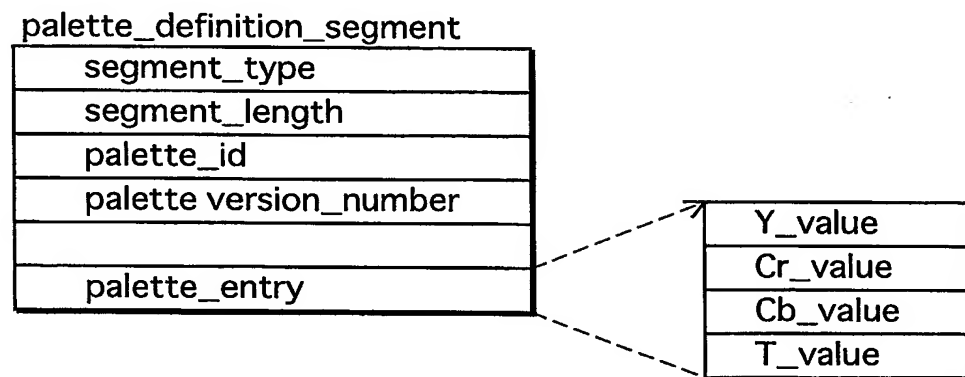


FIG.7B



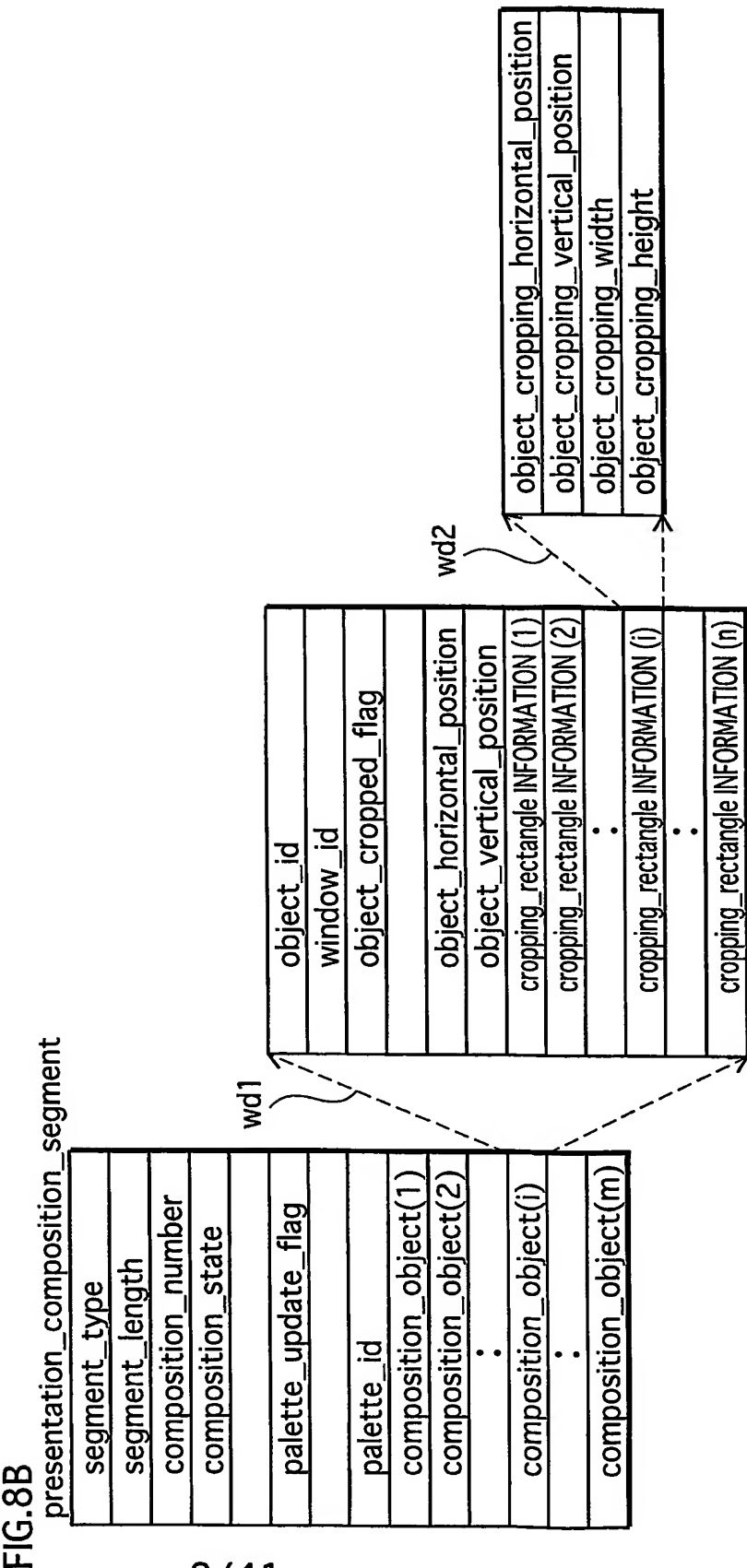




FIG.9

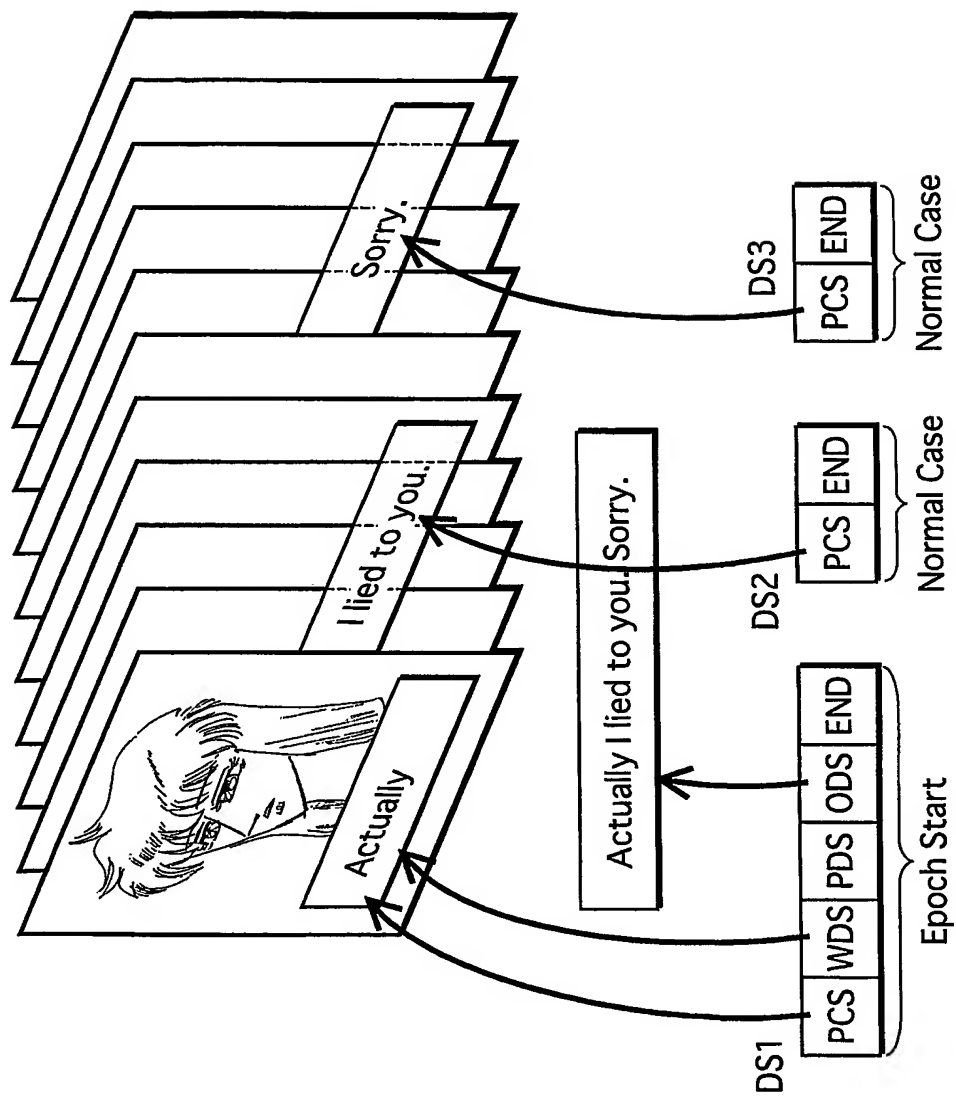


FIG.10

EXAMPLE DESCRIPTION  
OF PCS AND WDS IN DS1

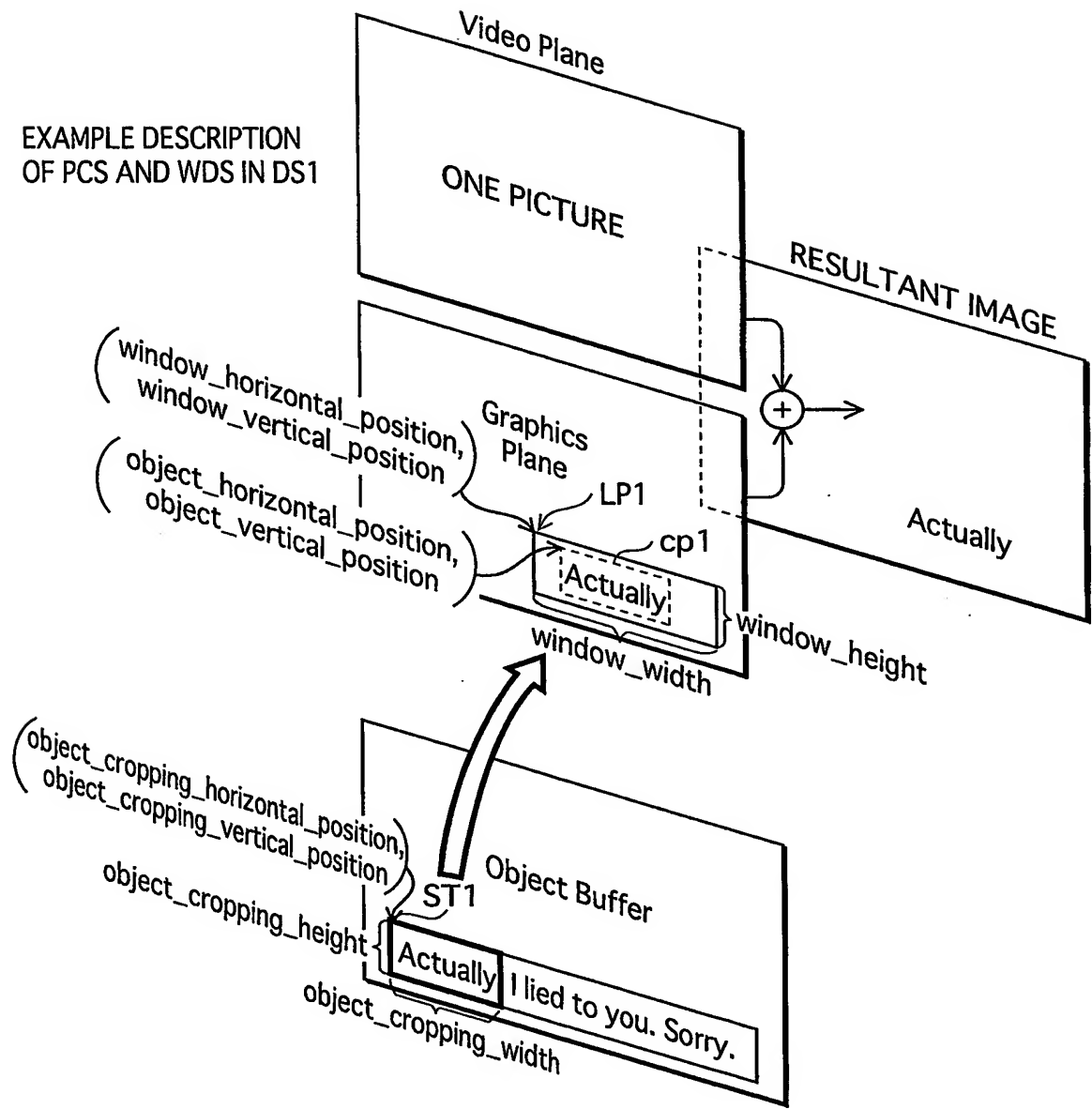


FIG. 11

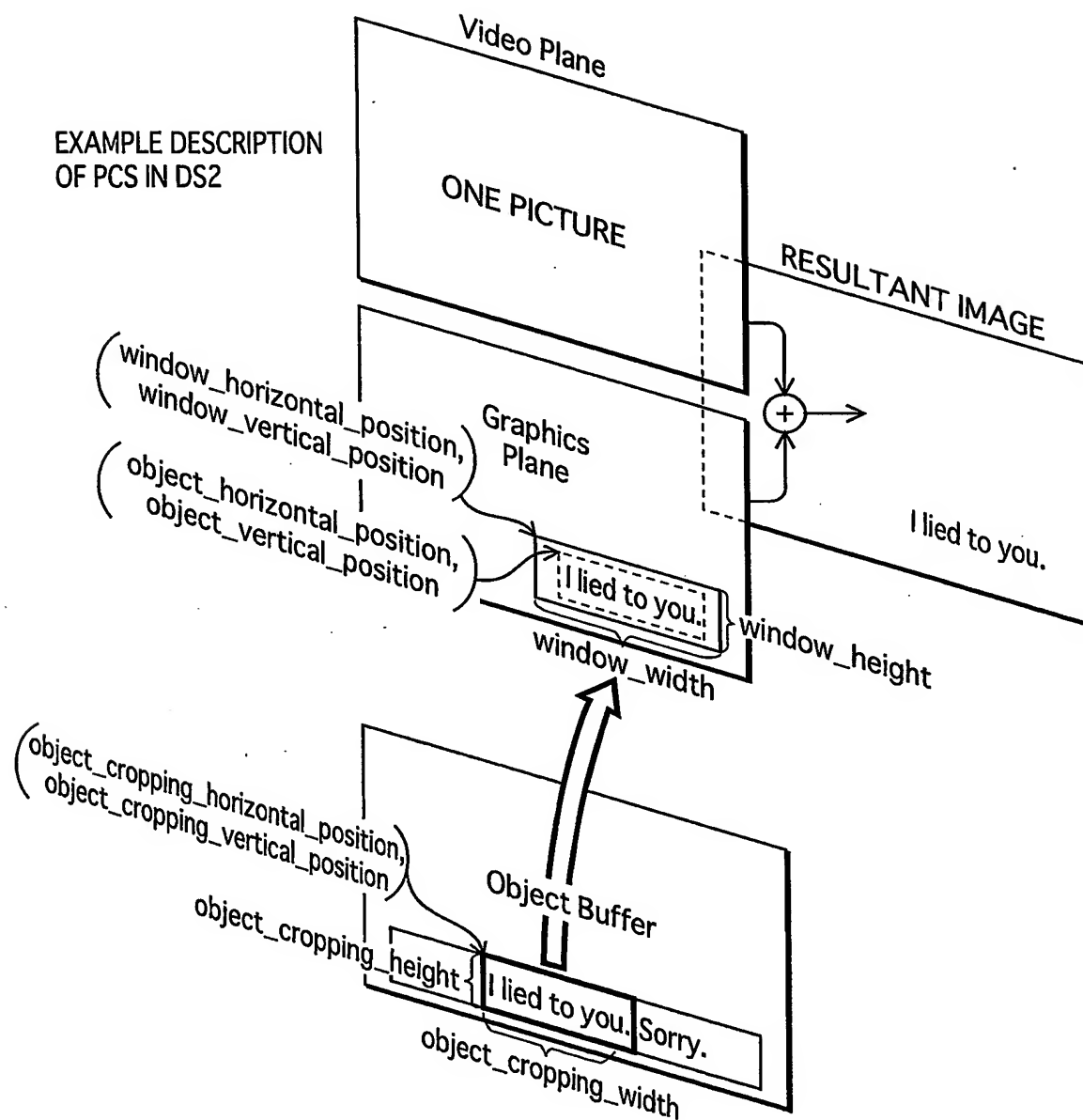
EXAMPLE DESCRIPTION  
OF PCS IN DS2

FIG. 12

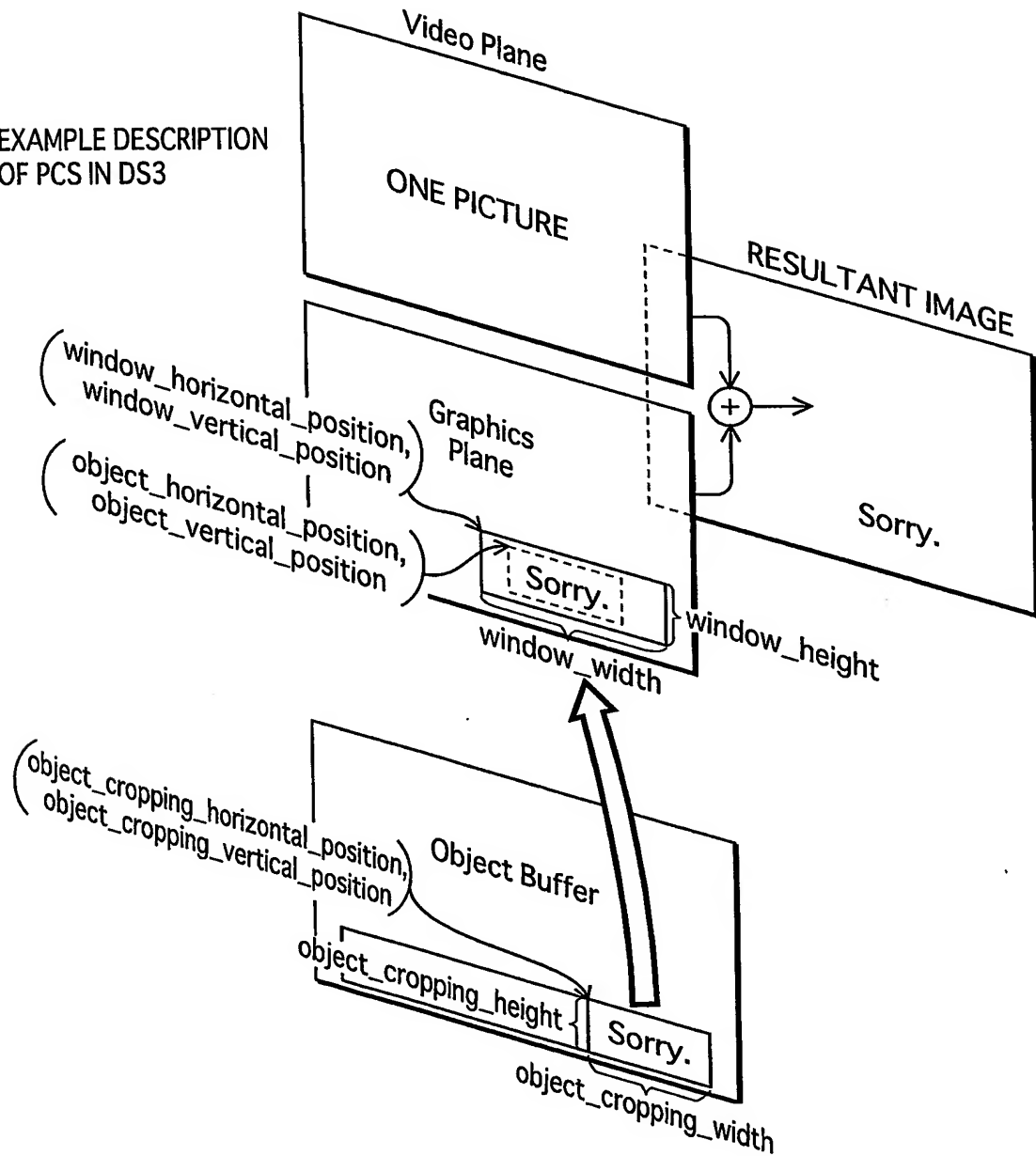
EXAMPLE DESCRIPTION  
OF PCS IN DS3

FIG.13

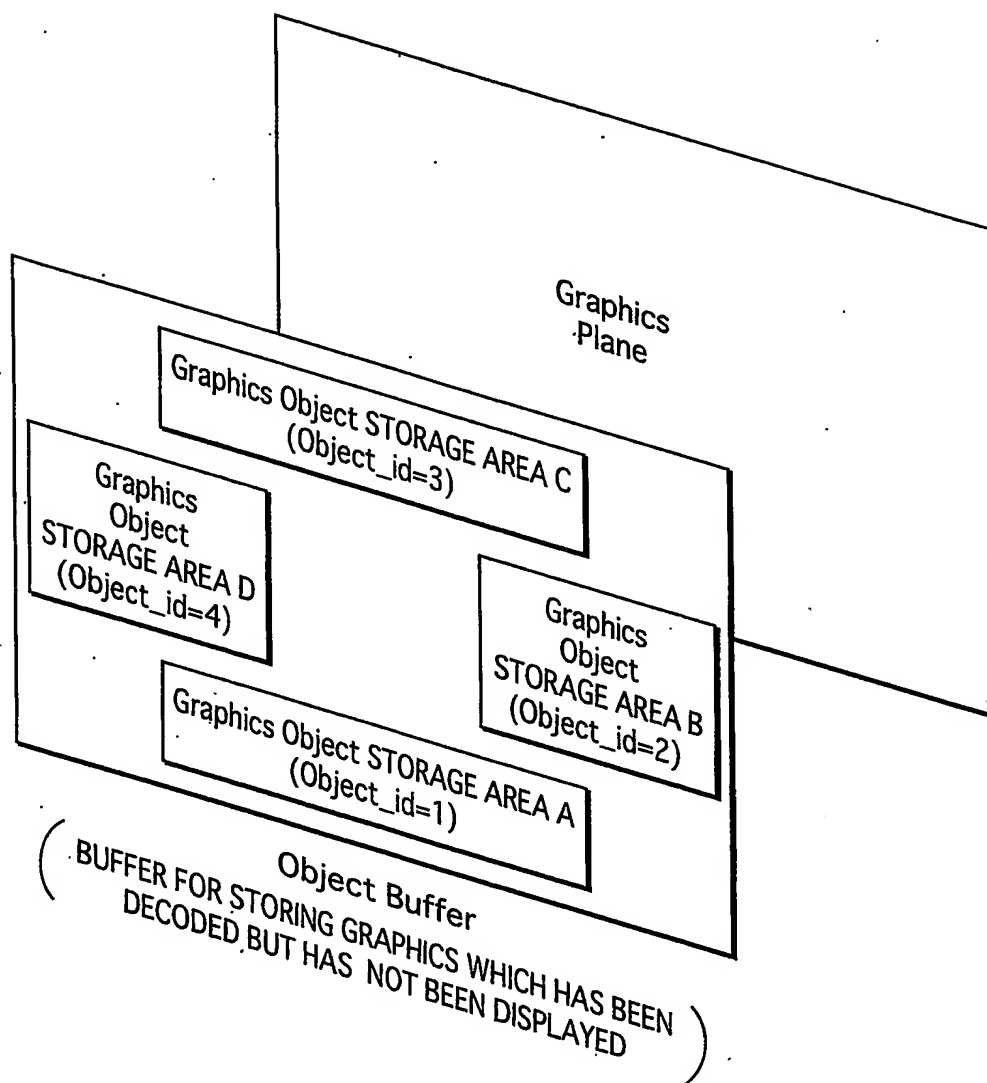


FIG. 14  $\text{PTS}(\text{DSn}[\text{PCS}]) \geq \text{DTS}(\text{DSn}[\text{PCS}]) + \text{DECODEDURATION}(\text{DSn})$

Where:

- $\text{DECODEDURATION}(\text{DSn})$  is calculated as follows:

```

decode_duration = 0 ;
decode_duration += PLANEINITIALIZATIONTIME( DSn ) ;
if( DSn. PCS. num_of_objects == 2 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    if( DSn. PCS. OBJ[0]. window_id == DSn. PCS. OBJ[1]. window_id )
    {
        decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration ) ;
        decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*106 ) ;
    }
    else
    {
        decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*106 ) ;
        decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration ) ;
        decode_duration += 90000*( SIZE( DSn. PCS. OBJ[1]. window_id )//256*106 ) ;
    }
}
else if( DSn. PCS. num_of_objects == 1 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*106 ) ;
}
return decode_duration ;

```

- $\text{PLANEINITIALIZATIONTIME}(\text{DSn})$  is calculated as follows:

```

initialize_duration=0 ;
if( DSn. PCS. composition_state == EPOCH_START )
{
    initialize_duration = 90000*( 8*video_width*video_height//256*106 ) ;
}
else
{
    for( i=0 ; i < WDS. num_windows ; i++ )
    {
        if( EMPTY(DSn.WDS.WIN[i],DSn ) )
            initialize_duration += 90000*( SIZE( DSn. WDS. WIN[i] )//256*106 ) ;
    }
}
return initialize_duration ;

```

- $\text{WAIT}(\text{DSn}, \text{OBJ}, \text{current\_duration})$  is calculated as follows:

```

wait_duration = 0 ;
if( EXISTS( OBJ. object_id, DSn ) )
{
    object_definition_ready_time = PTS( GET( OBJ. object_id, DSn ) ) ;
    current_time = DTS( DSn. PCS )+current_duration ;
    if( current_time < object_definition_ready_time )
        wait_duration += object_definition_ready_time - current_time ;
}
return wait_duration ;

```

FIG. 15

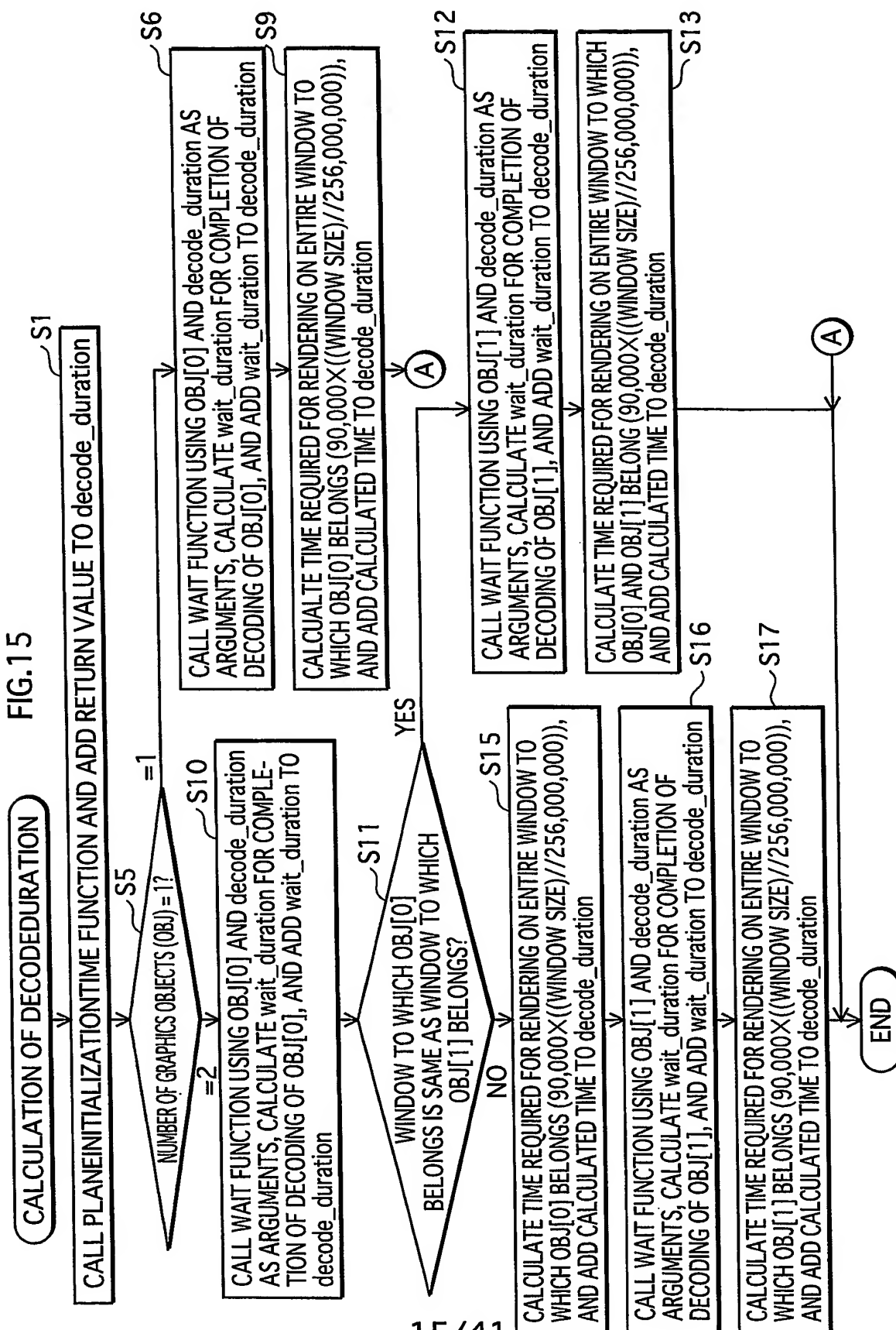


FIG. 1 6A

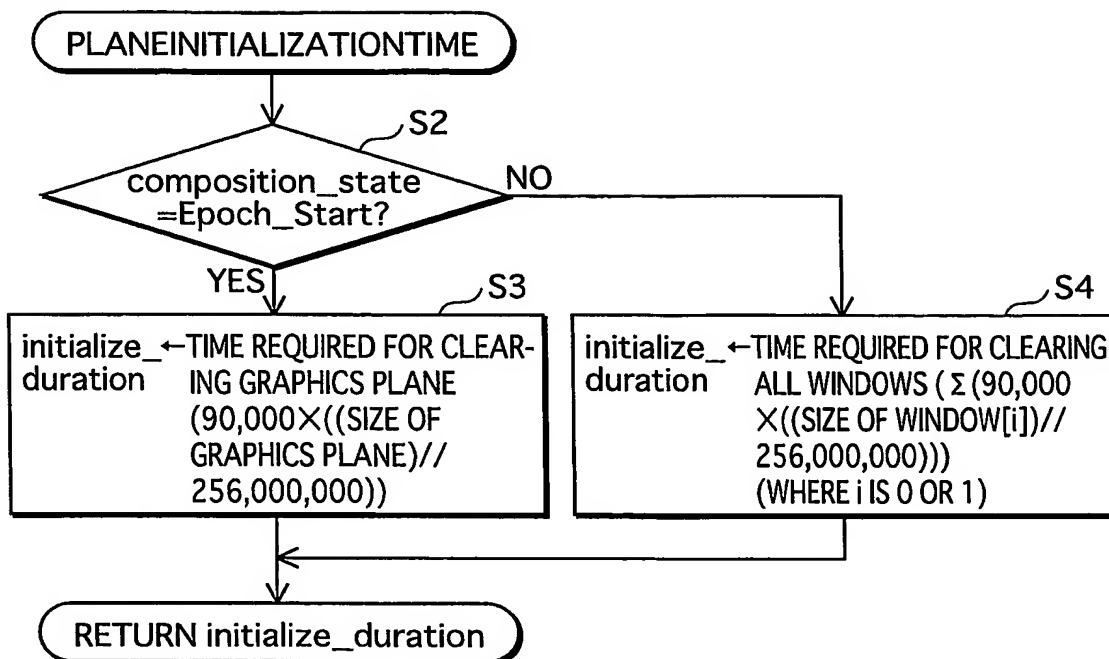


FIG. 1 6B

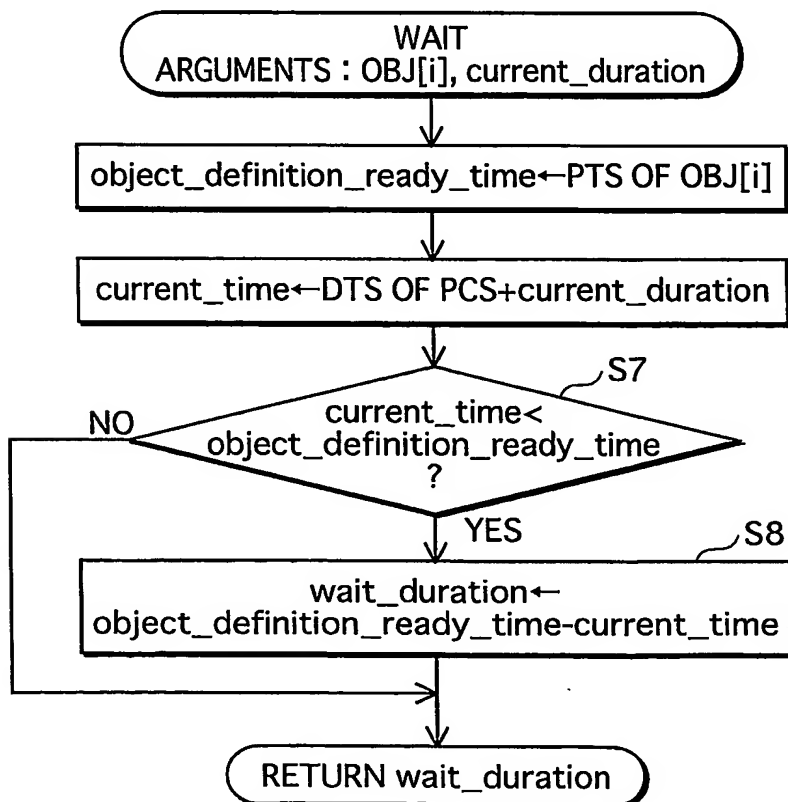




FIG.17A

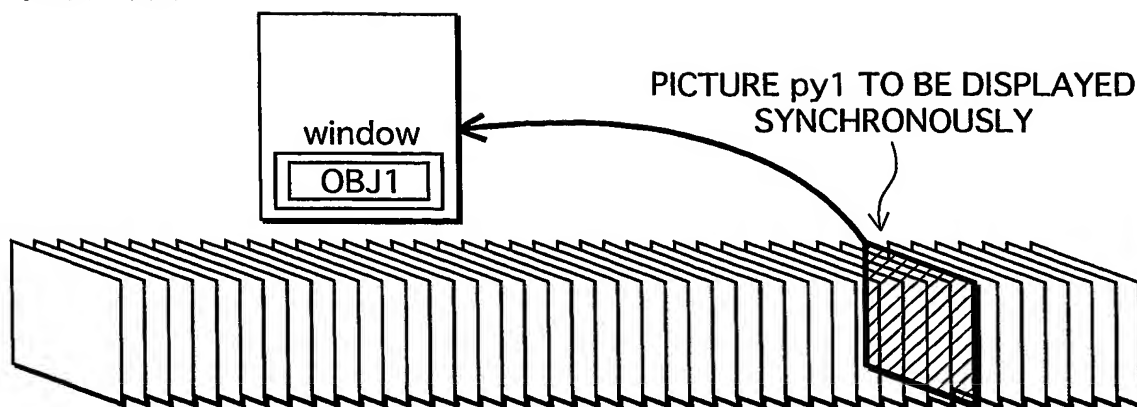


FIG.17B

DECODE\_DURATION  
=(2)+(3)

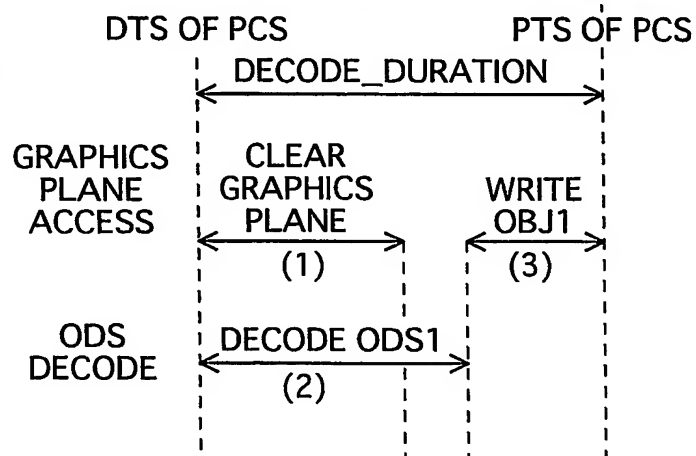


FIG.17C

DECODE\_DURATION  
=(1)+(3)

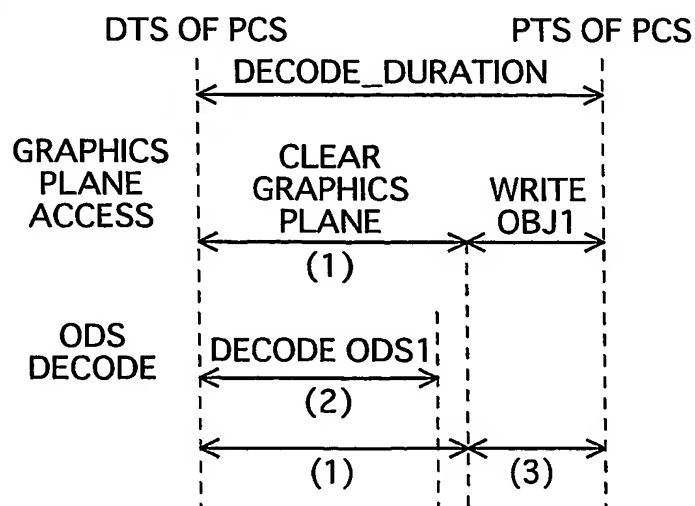


FIG.18A

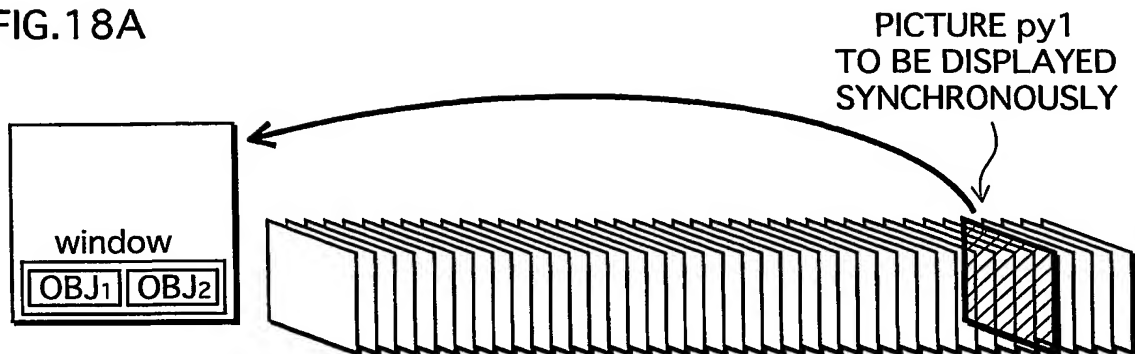


FIG.18B

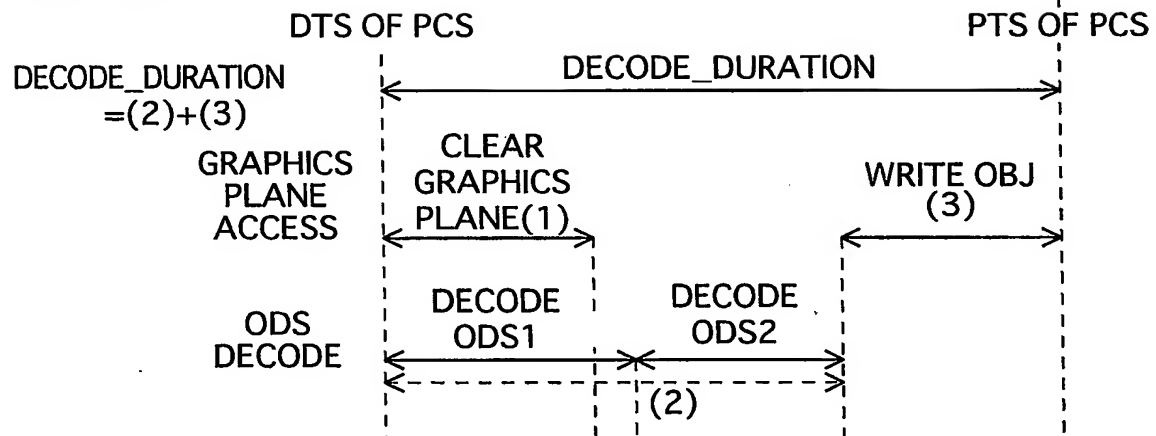
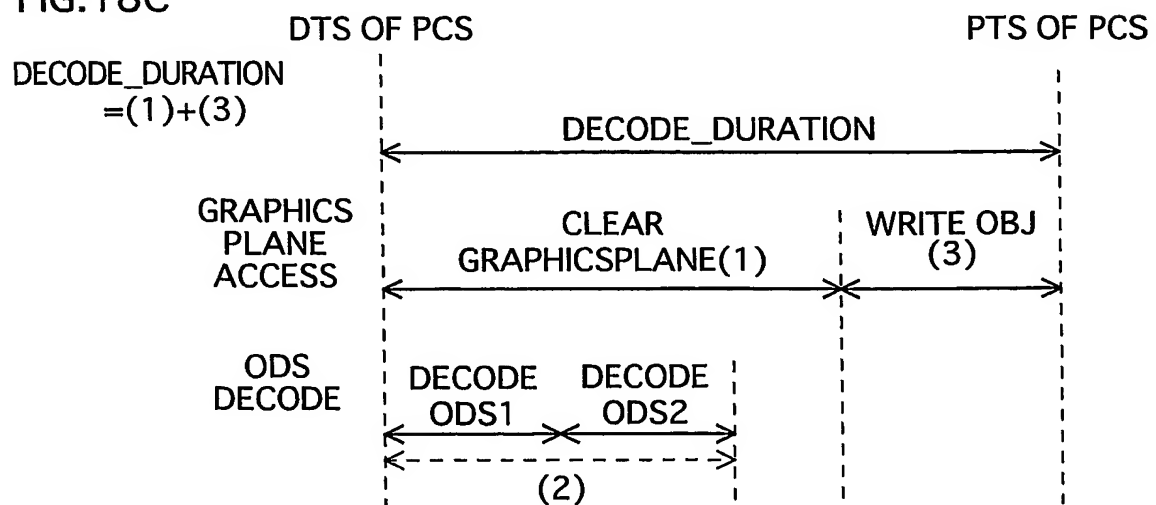


FIG.18C



PICTURE py1 TO BE DISPLAYED  
SYNCHRONOUSLY

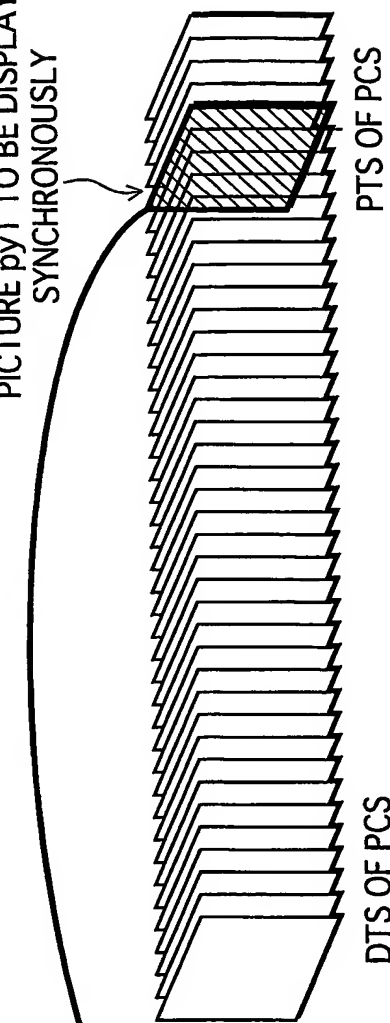


FIG.19A

FIG.19B

DECODE\_DURATION  
=(2)+(32)

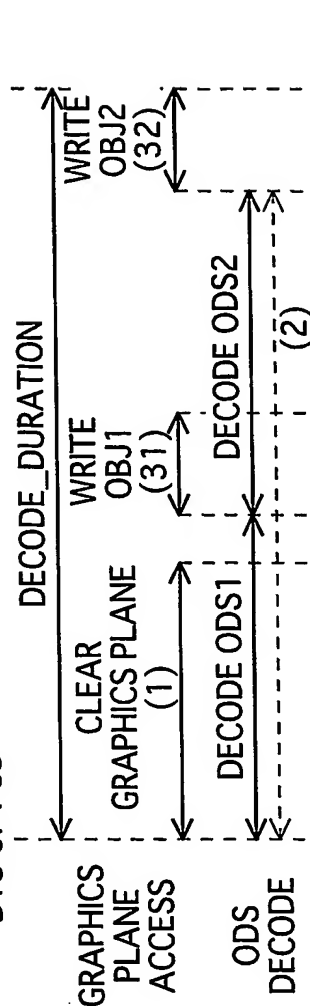


FIG.19C

DECODE\_DURATION  
=(1)+(31)+(32)

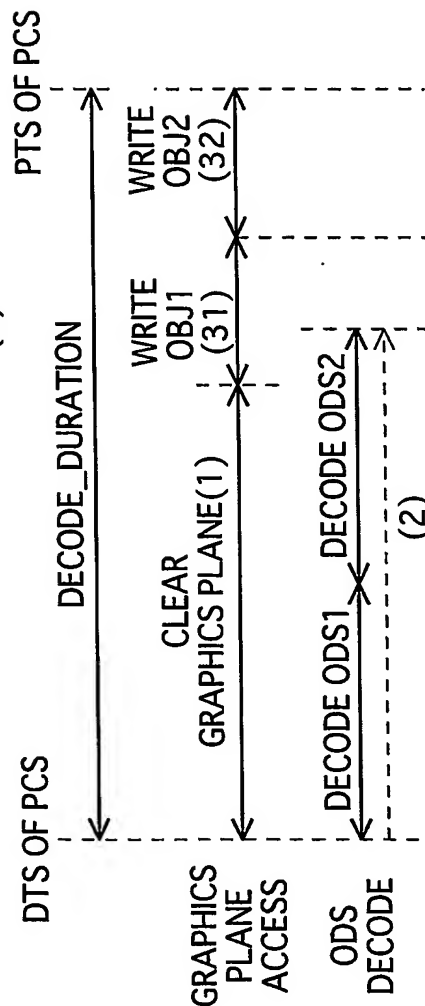


FIG.20

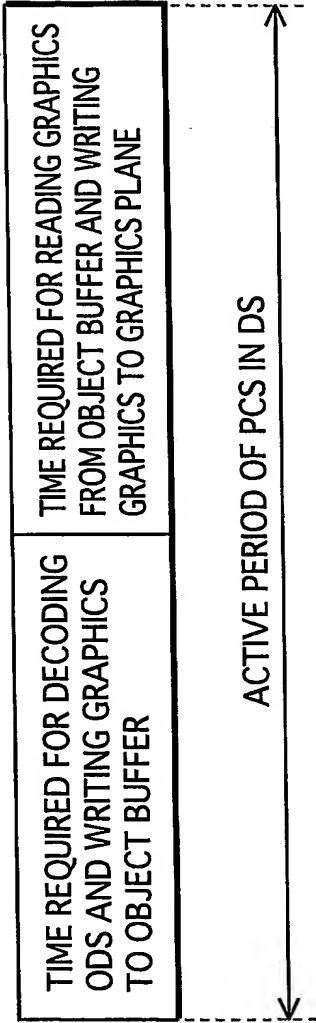


FIG.21

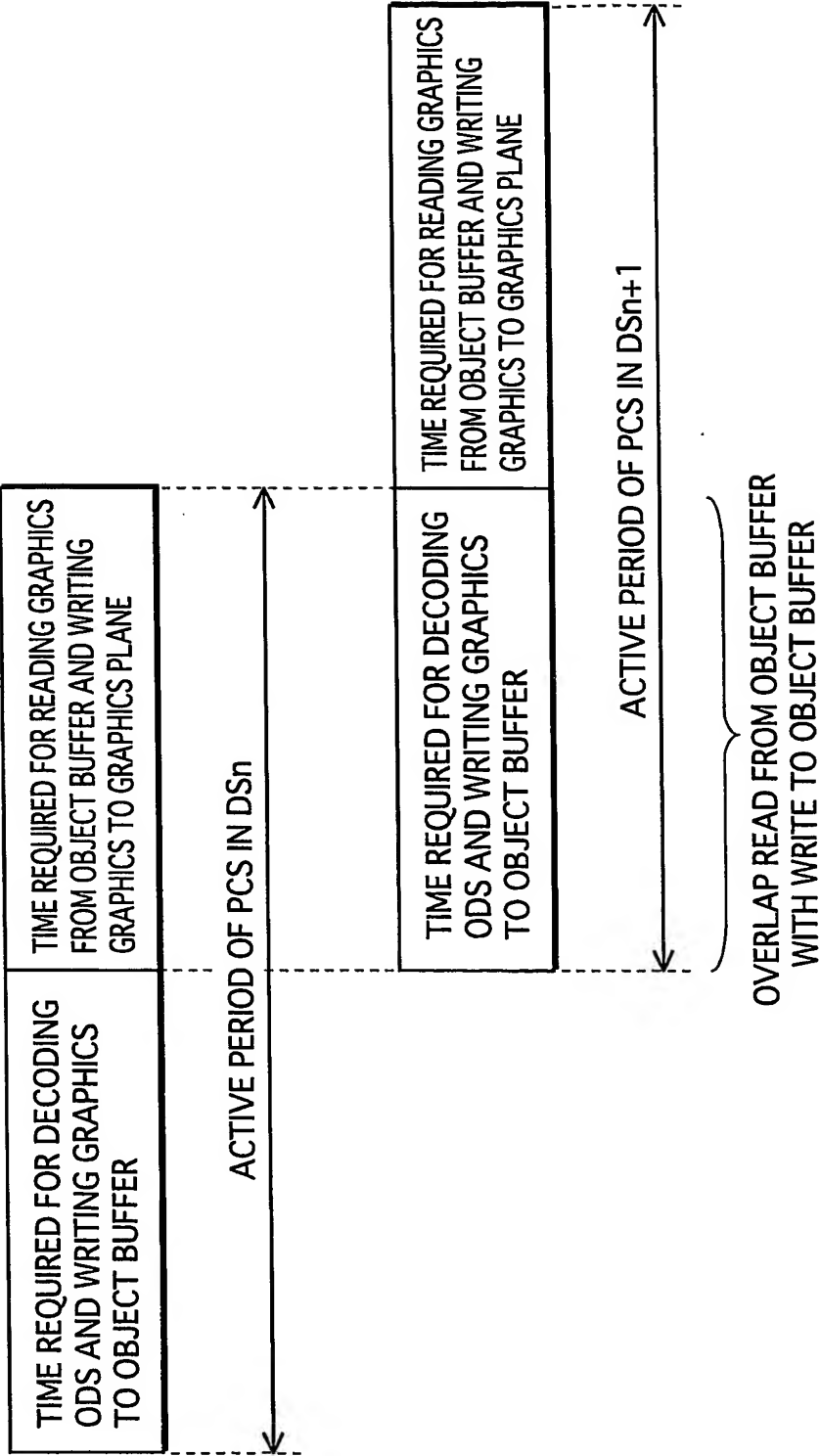


FIG.22

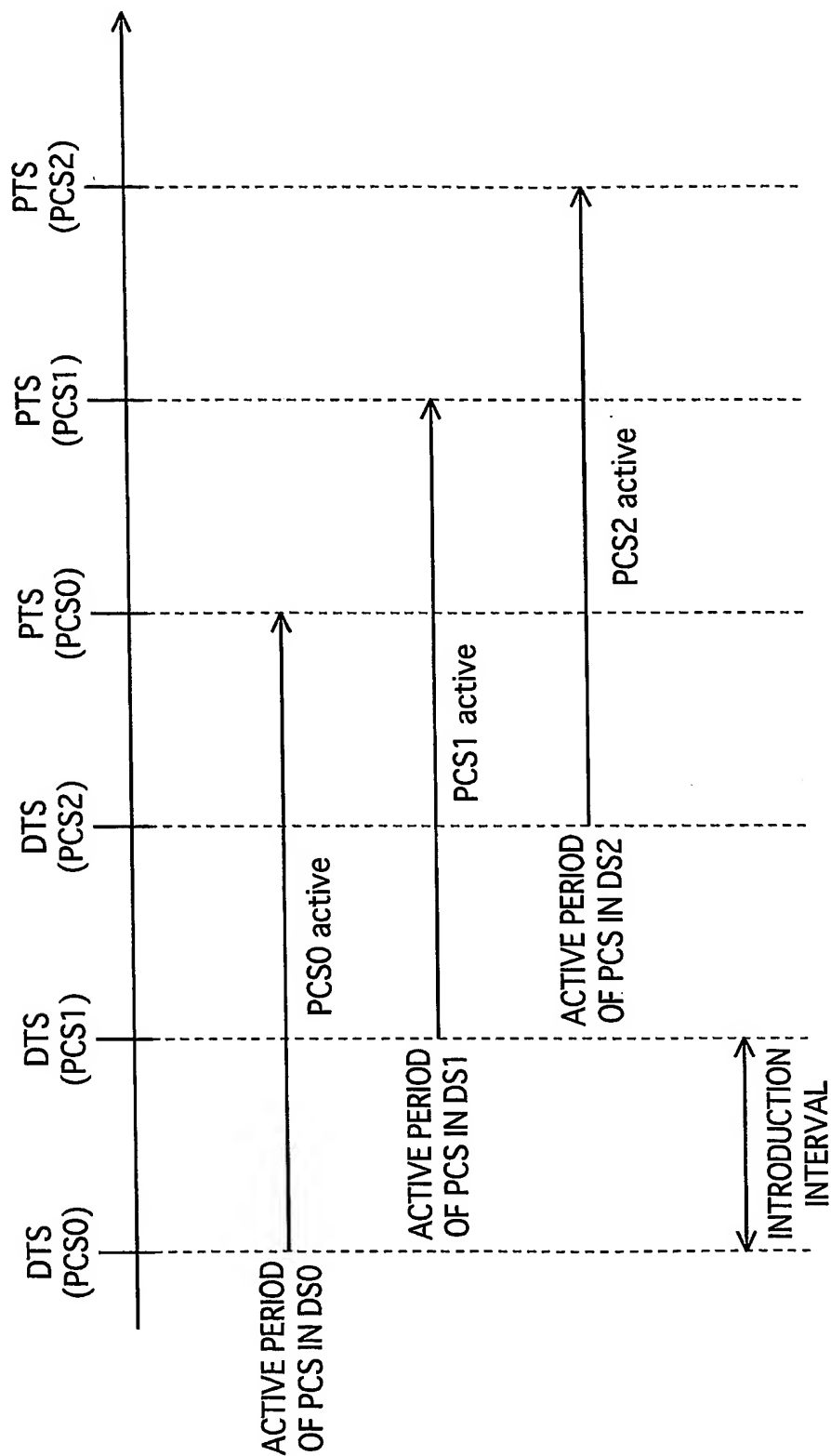


FIG. 23

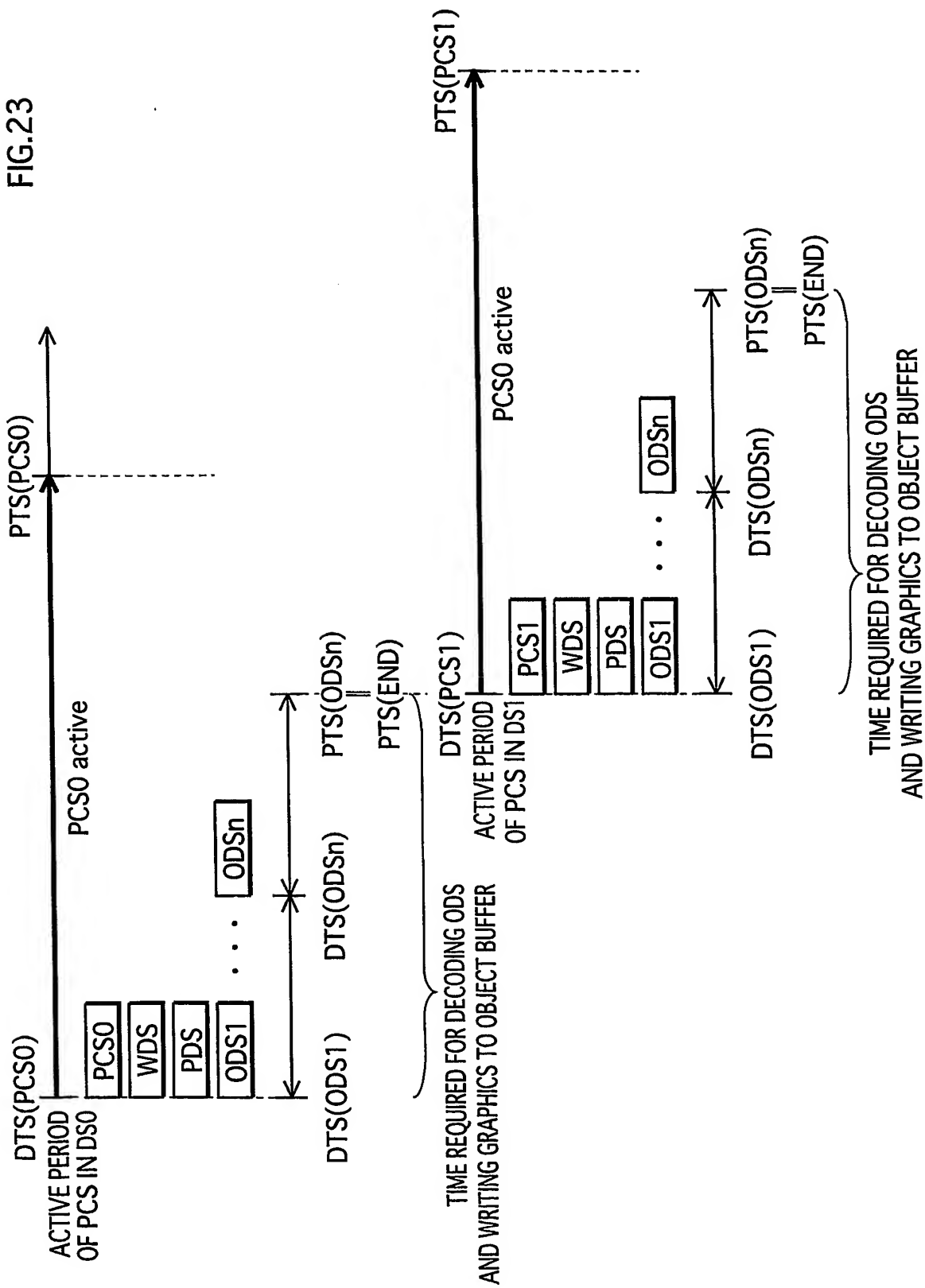


FIG.24

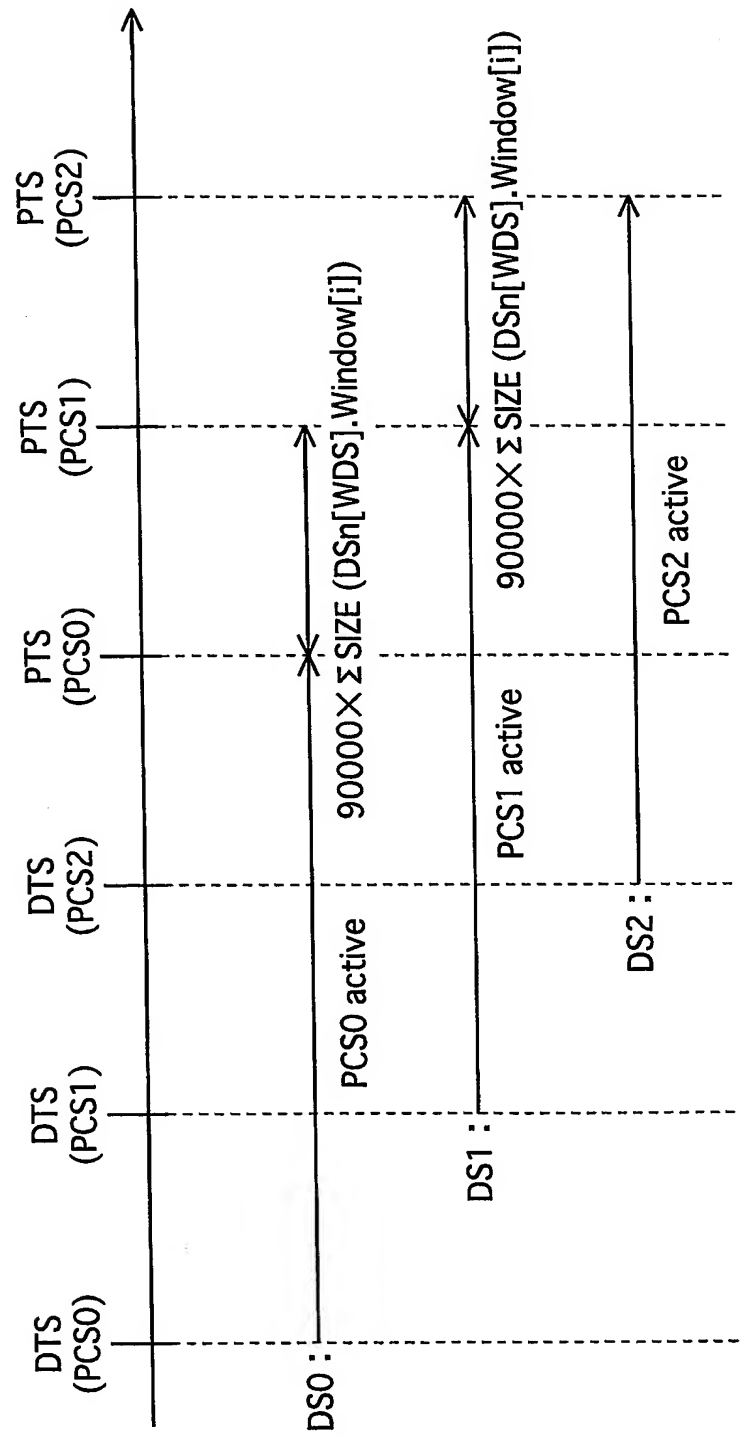
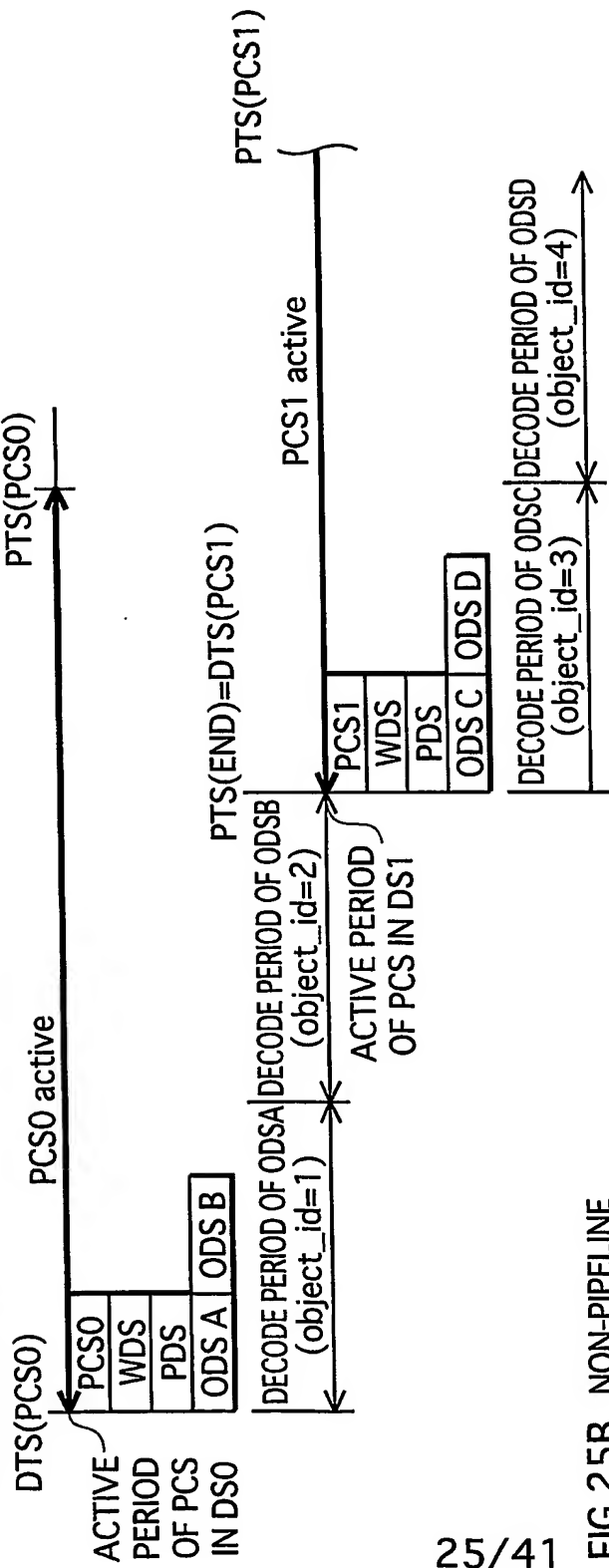




FIG.25A PIPELINE



25/41

FIG.25B NON-PIPELINE

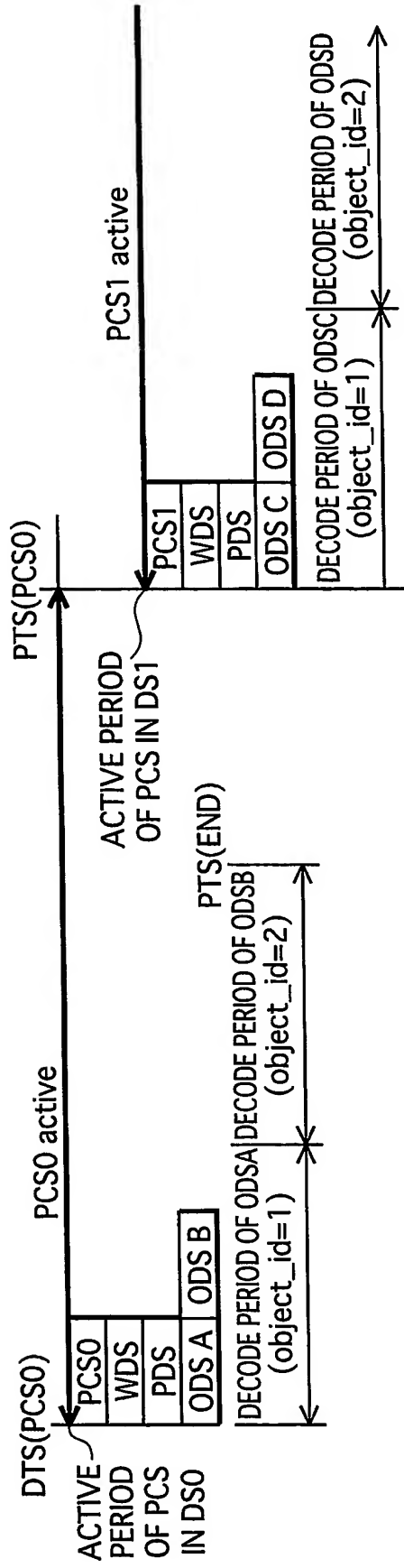


FIG.26

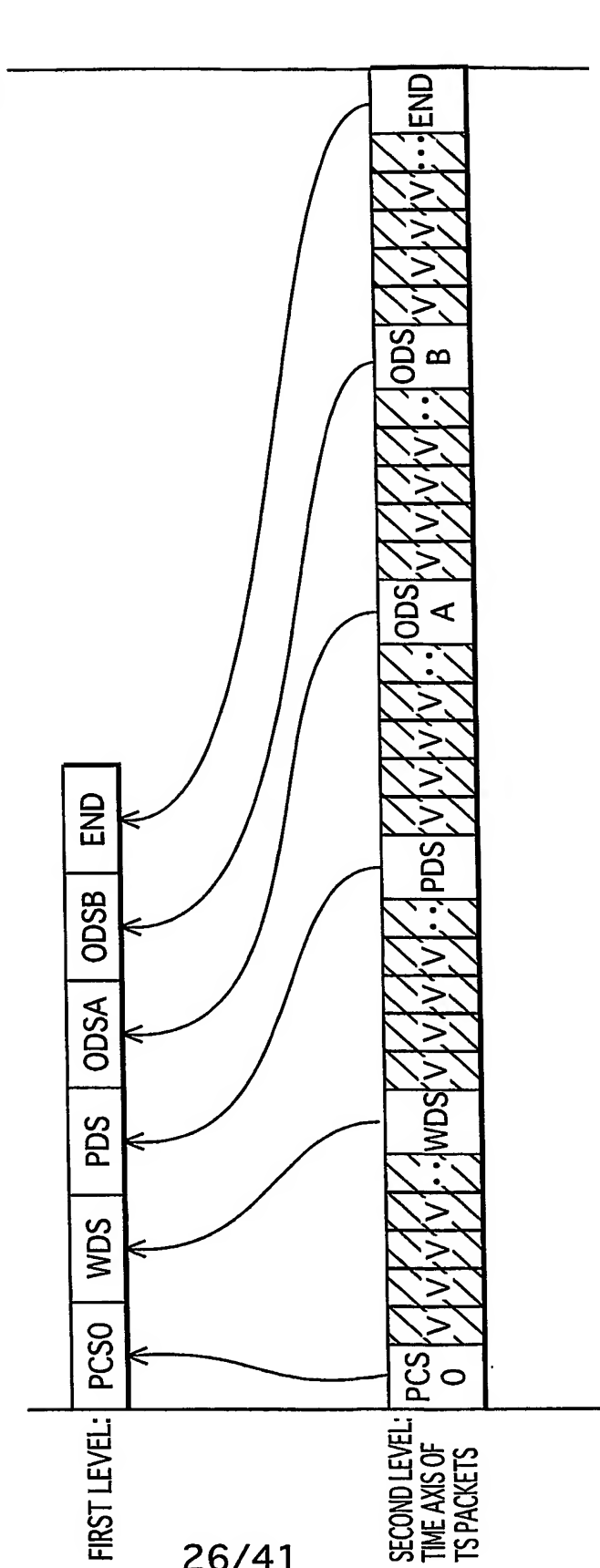


FIG.27A SCREEN COMPOSITION

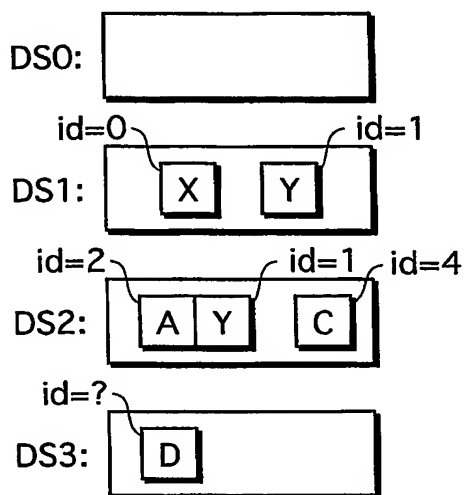


FIG.27B ACTIVE PERIOD OVERLAPPING AND ODS TRANSFER

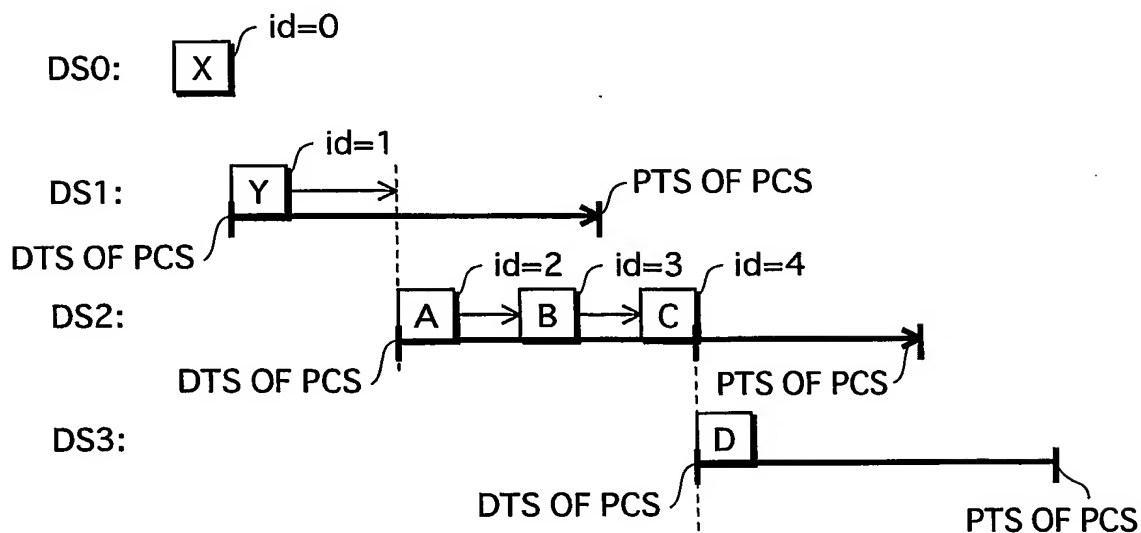


FIG.27C ARRANGEMENT IN OBJECT BUFFER

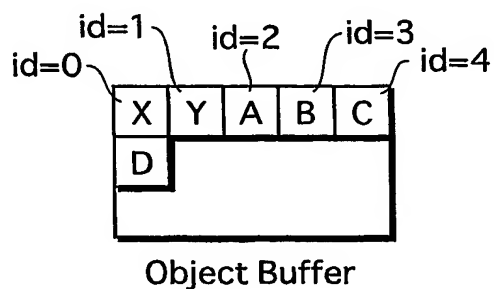


FIG.28

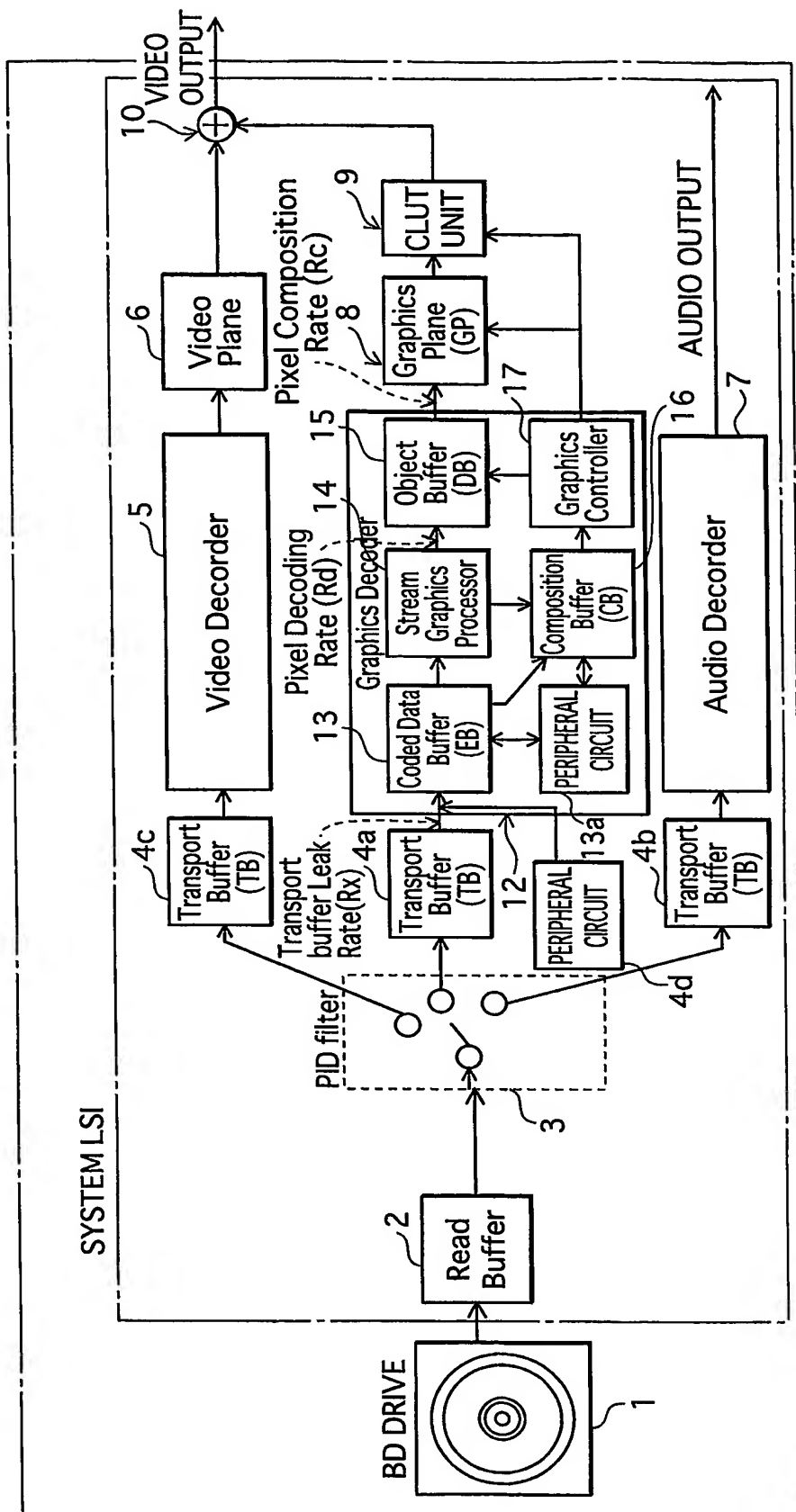
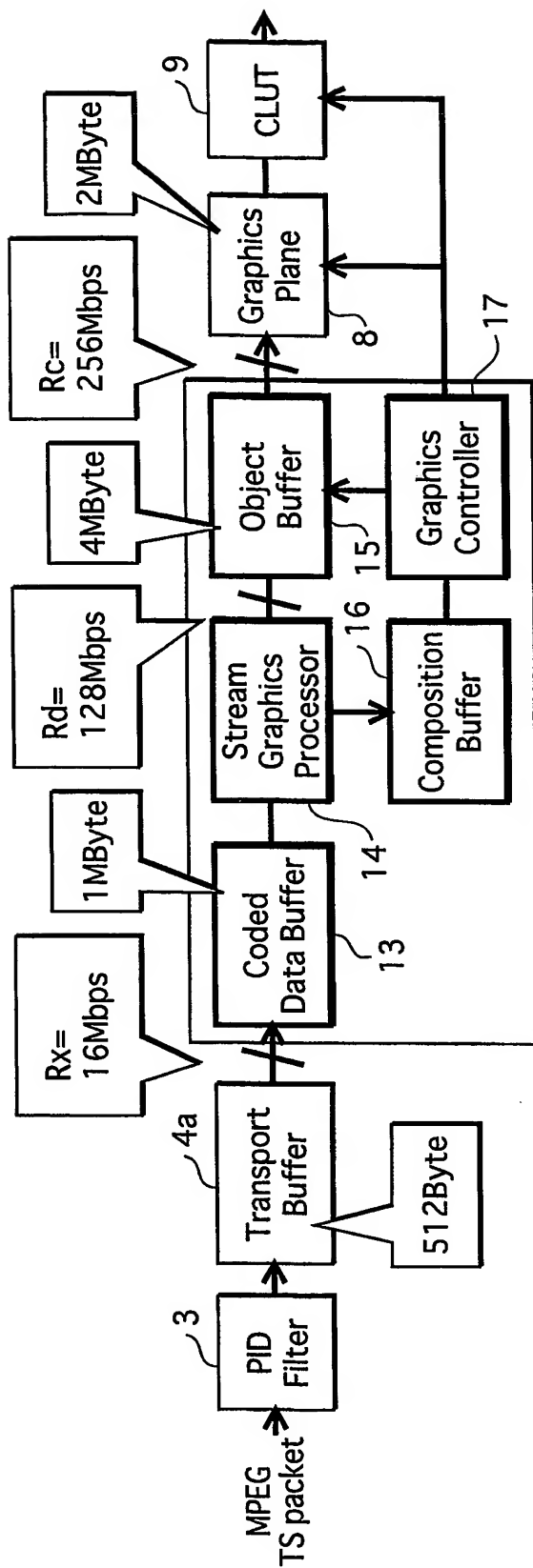
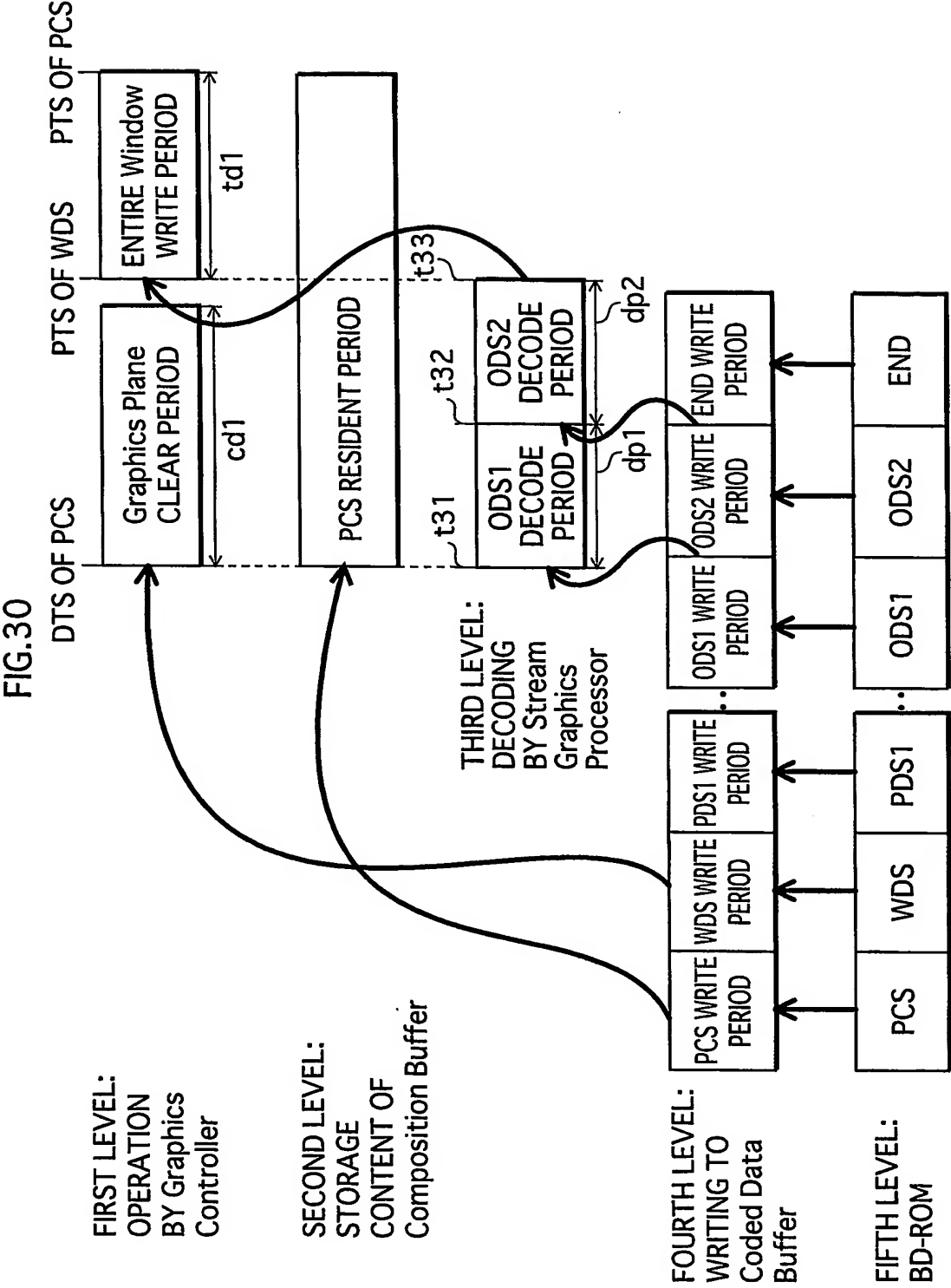
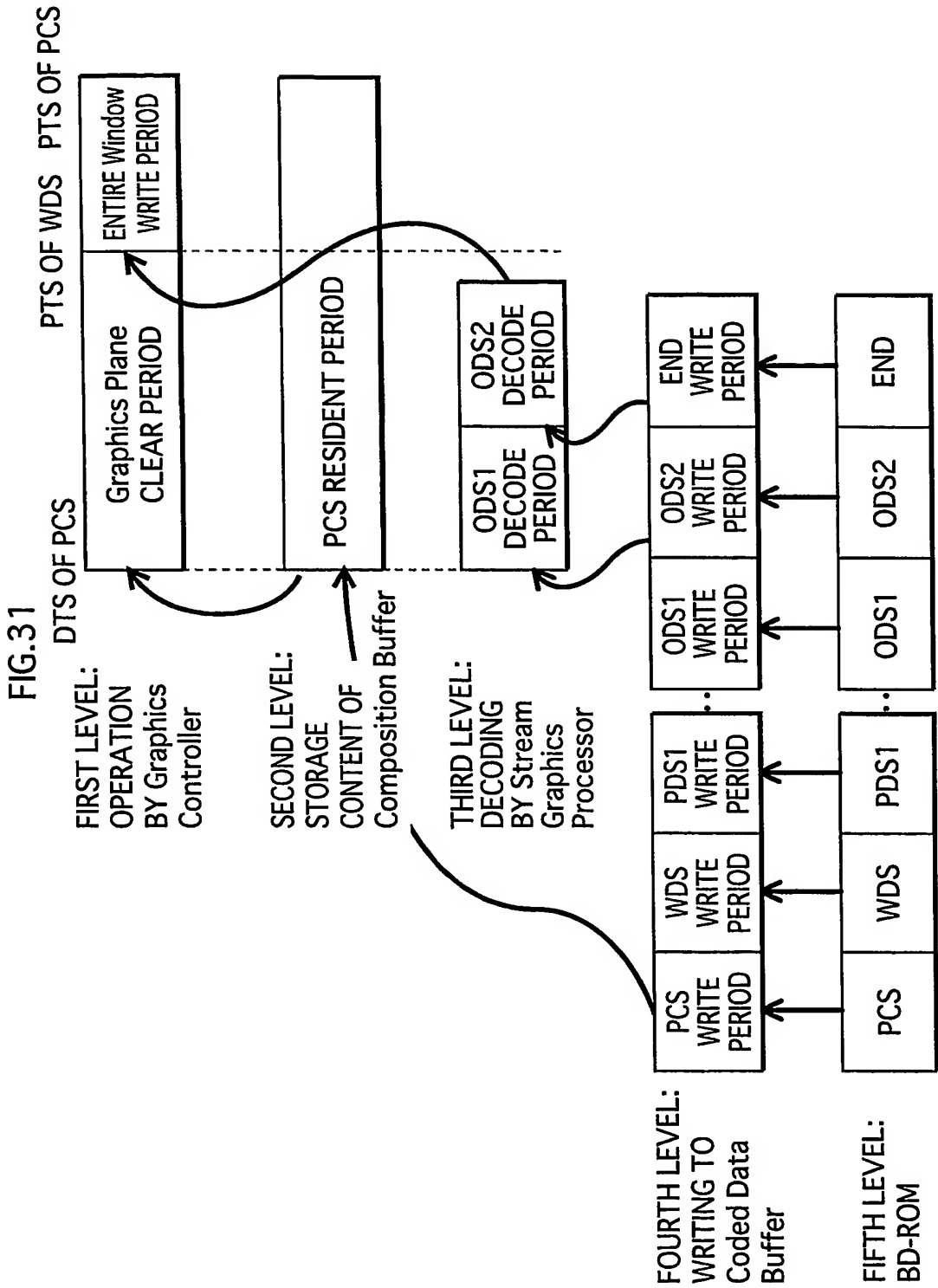


FIG.29







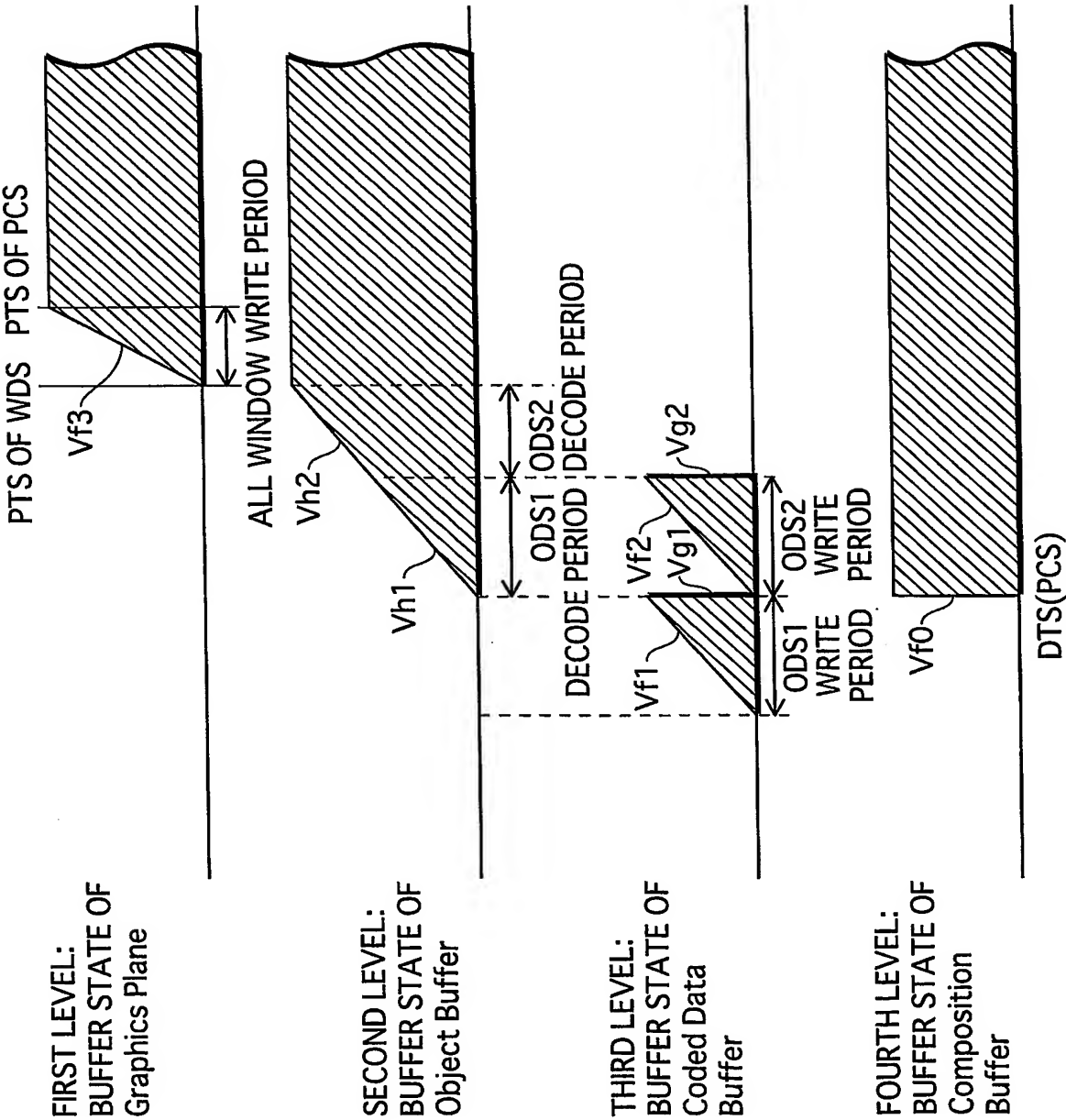


FIG.32



FIG.33

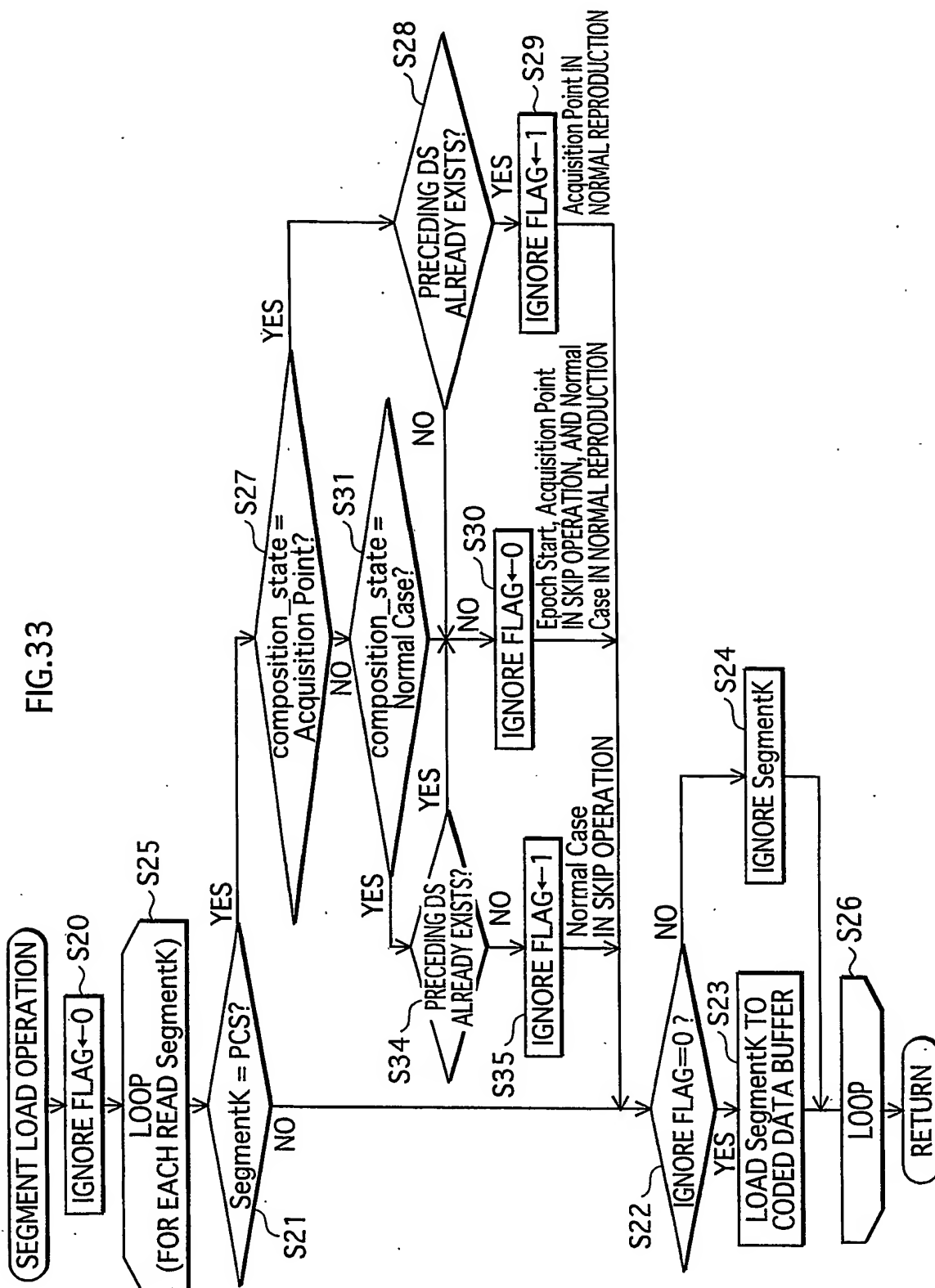


FIG.34

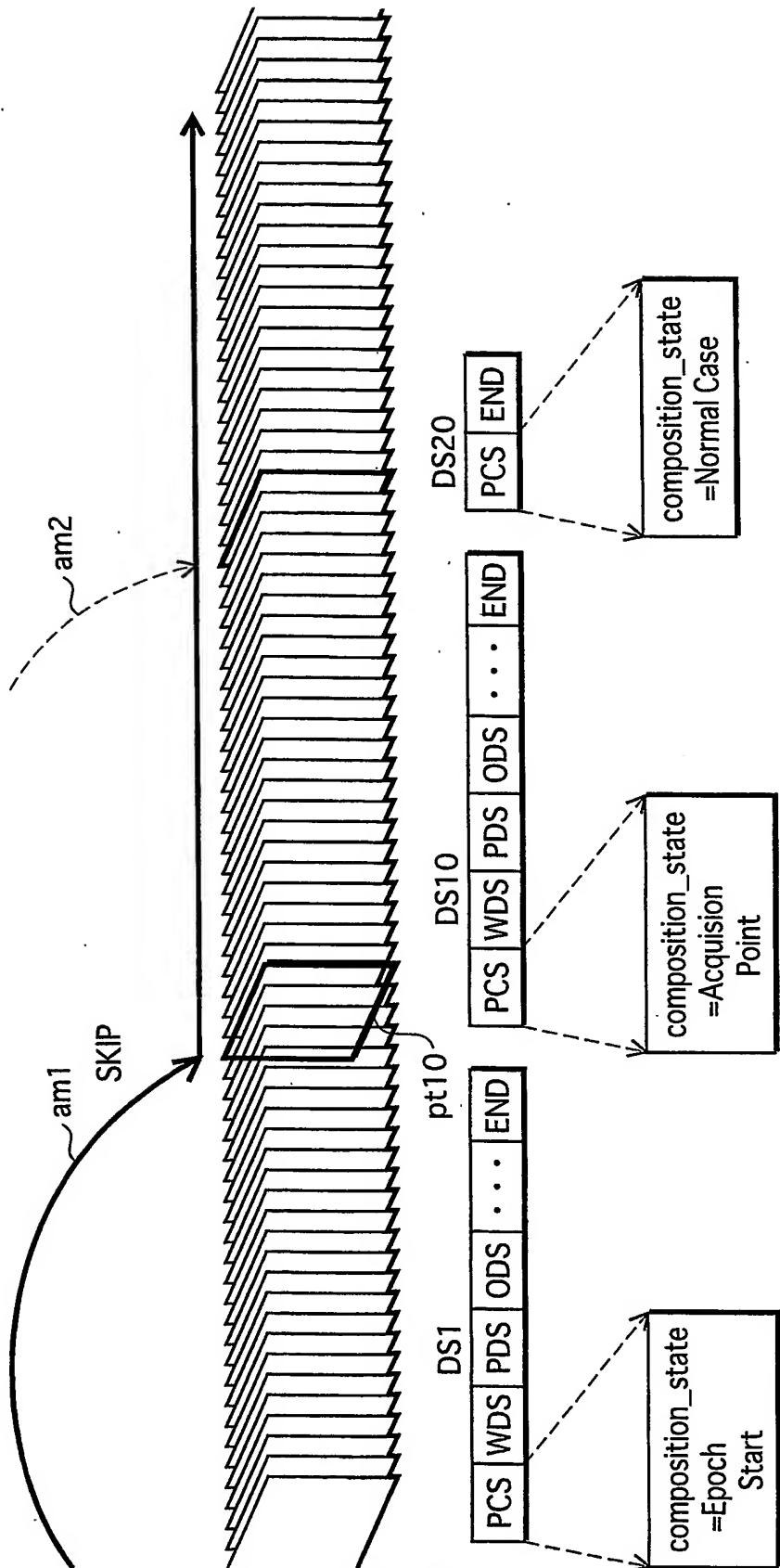


FIG.35

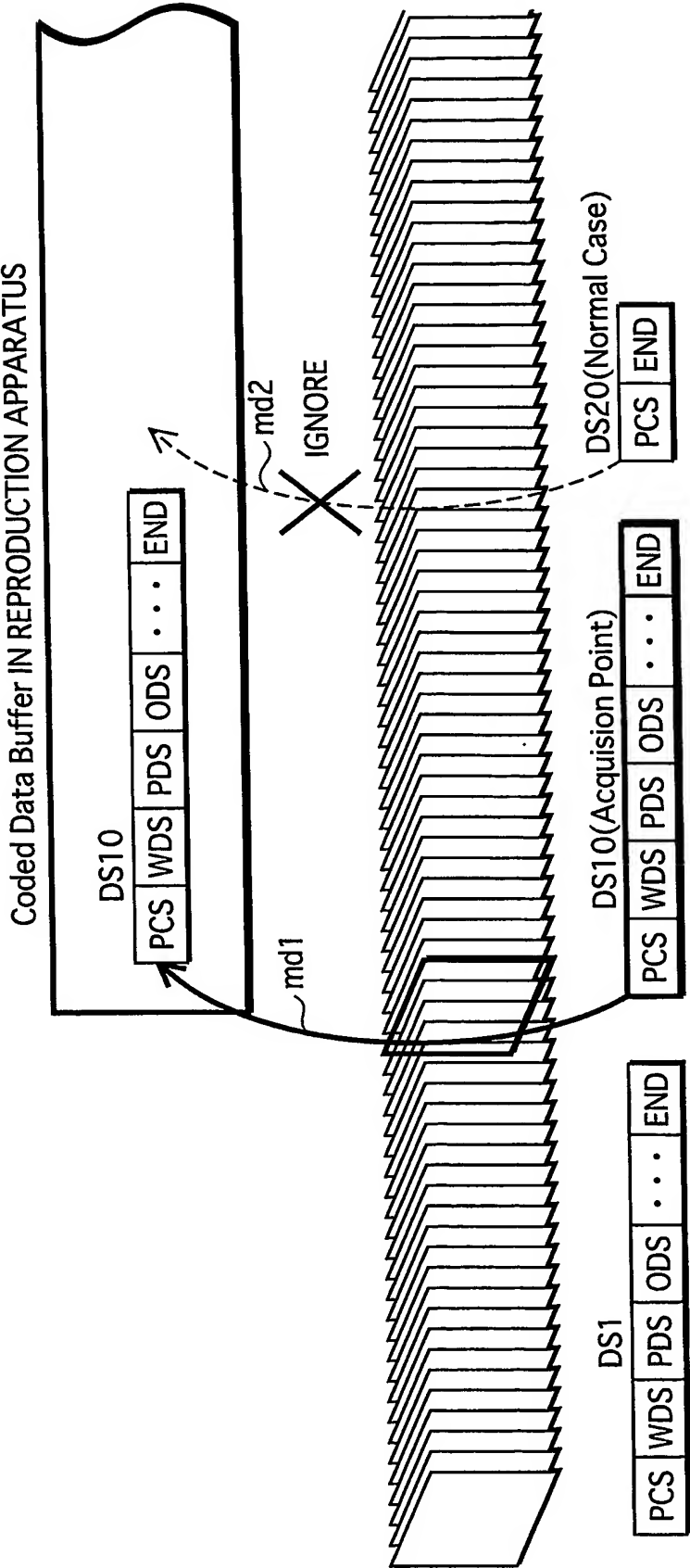


FIG.36

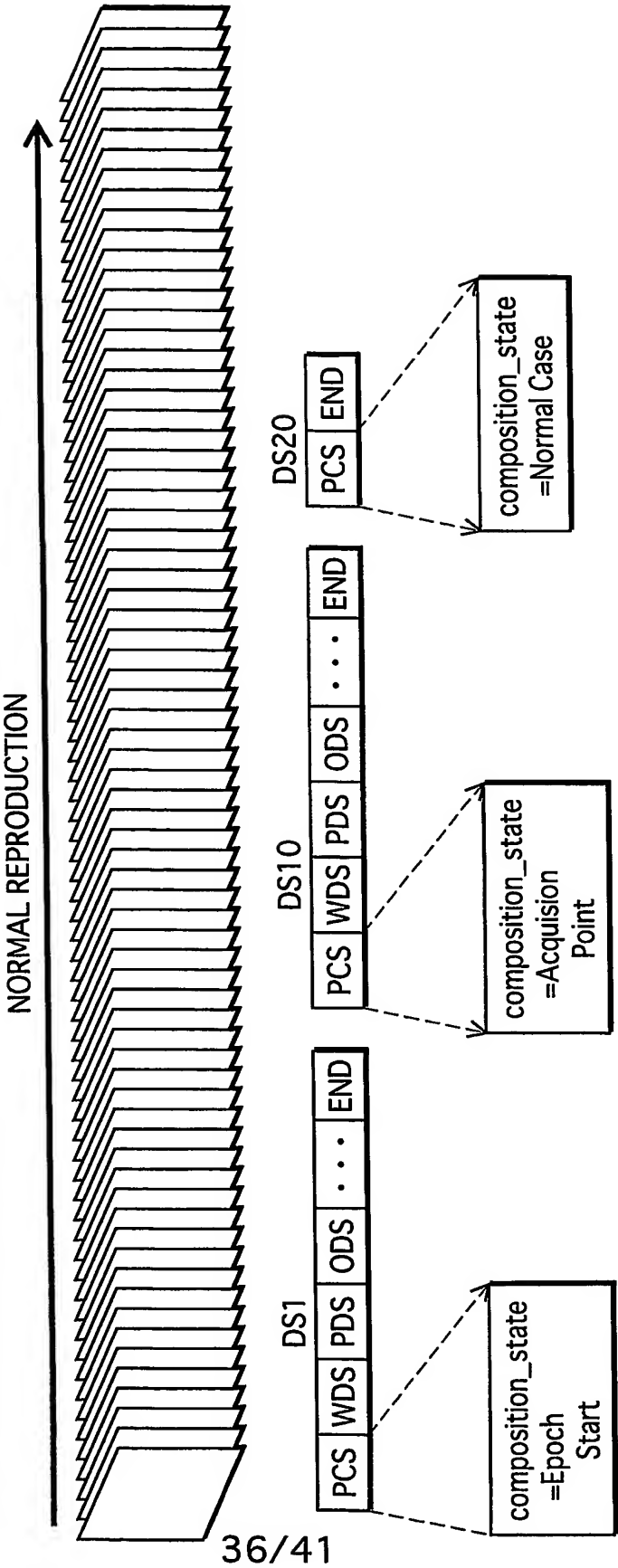


FIG.37

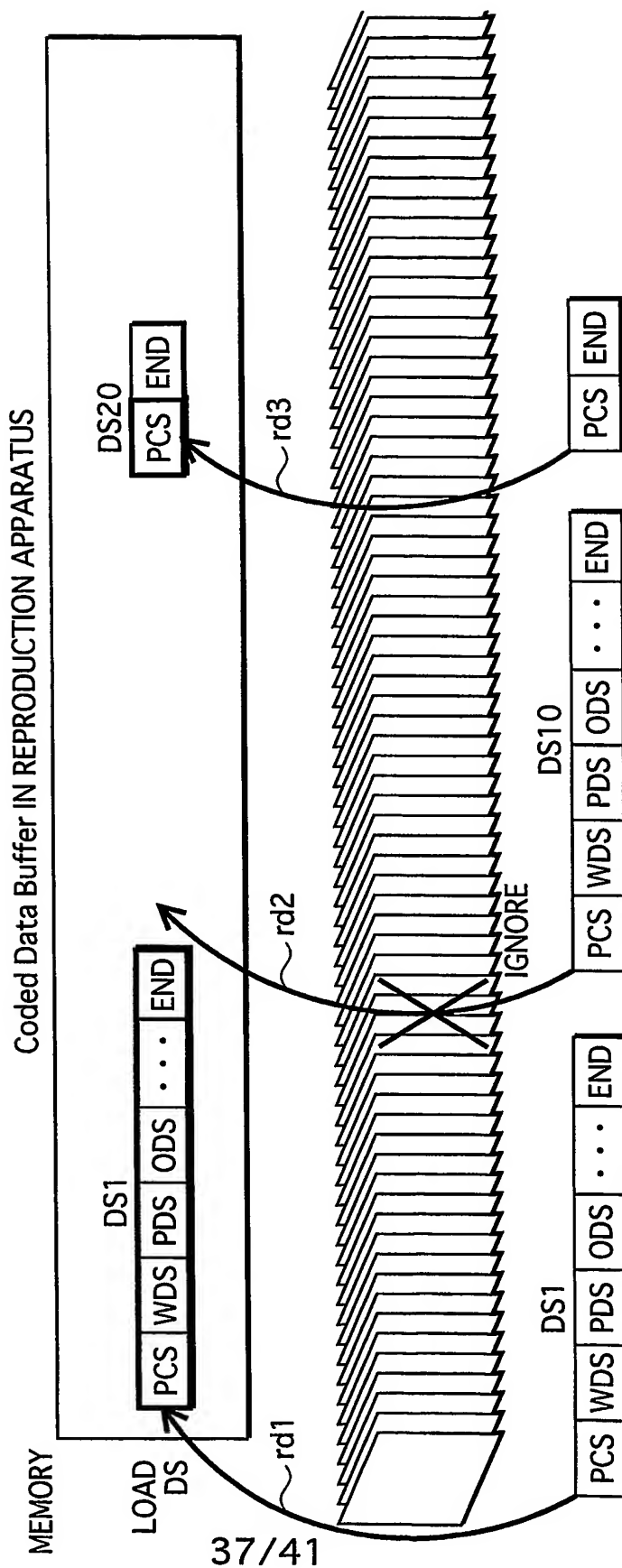


FIG. 38

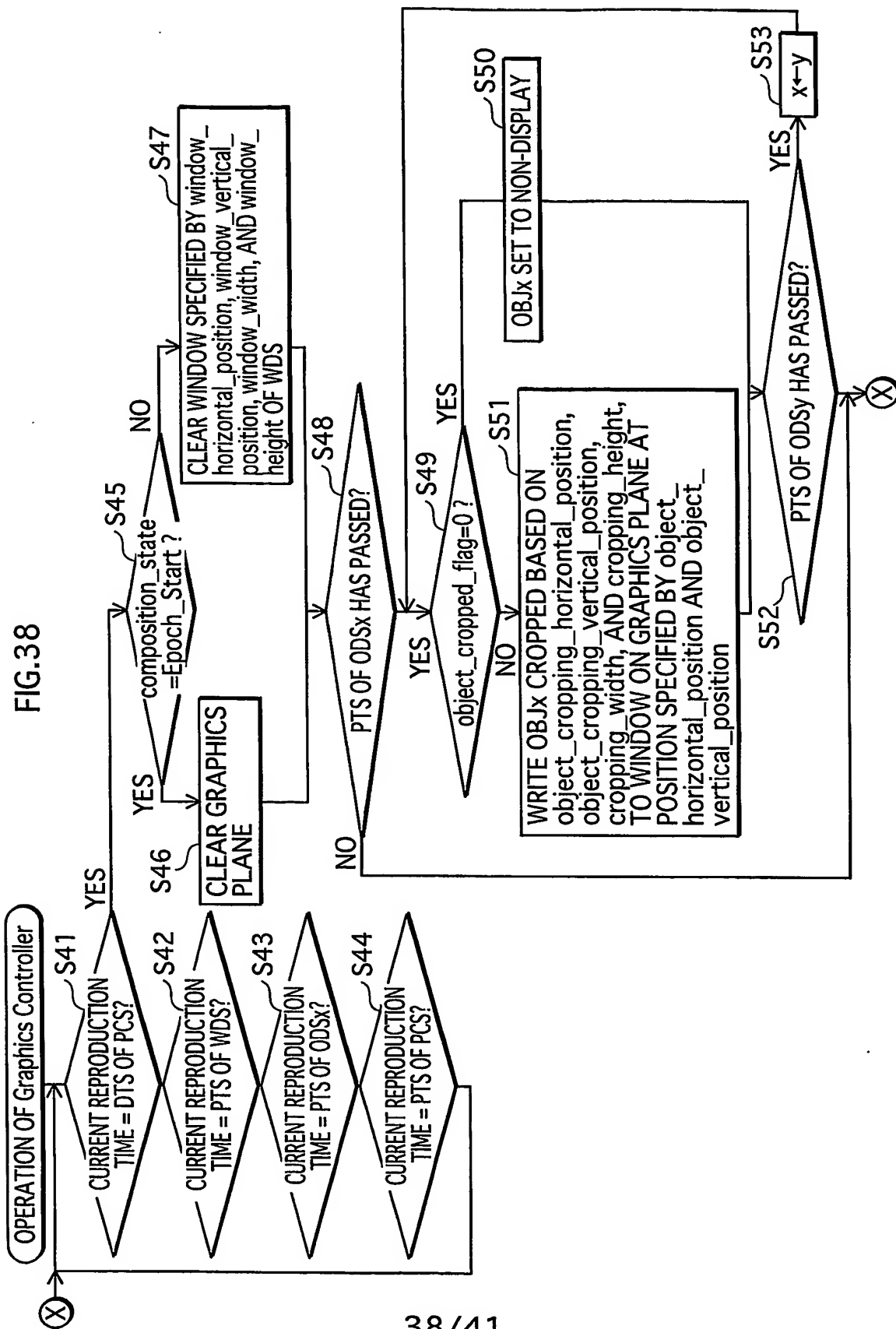


FIG.39

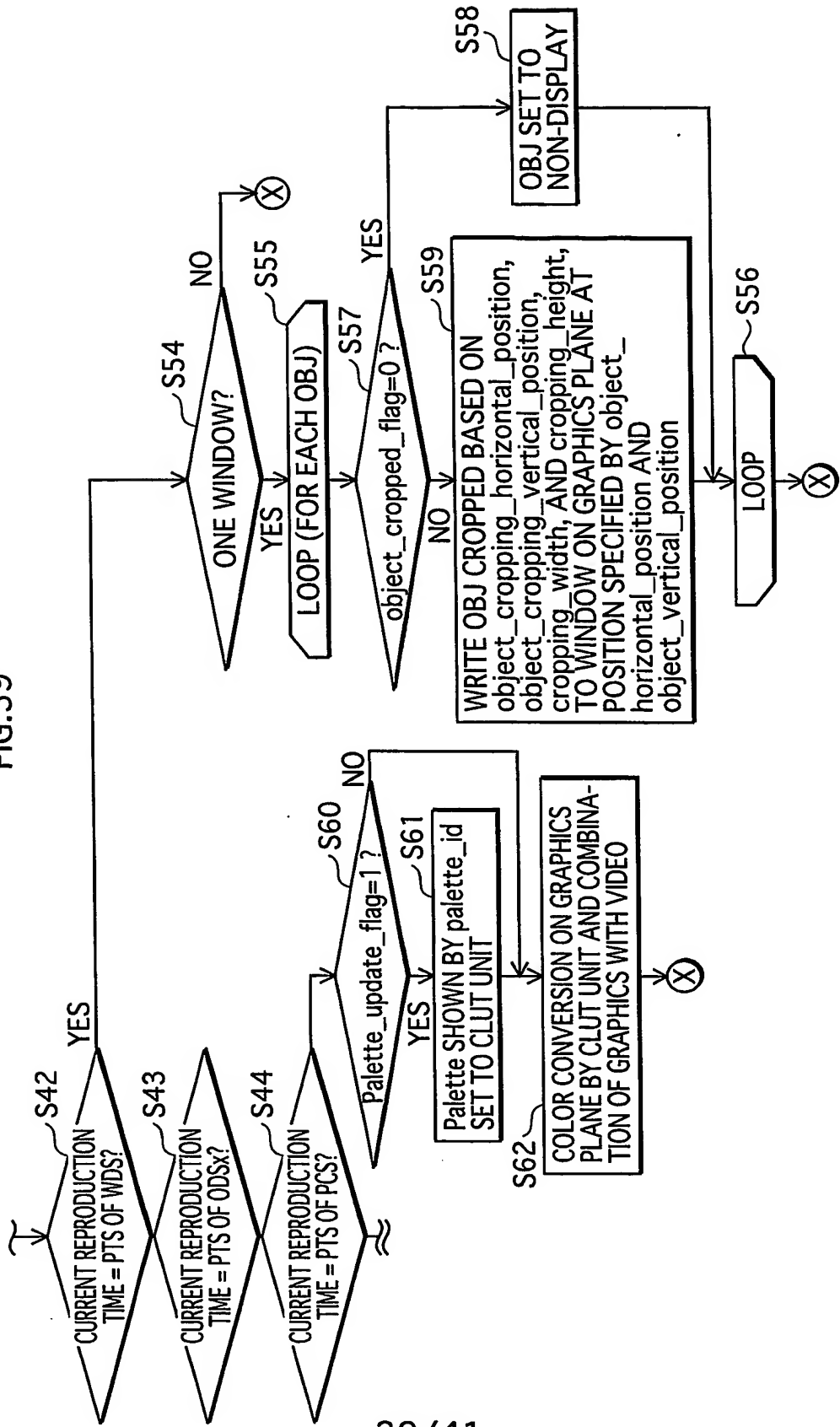


FIG.40

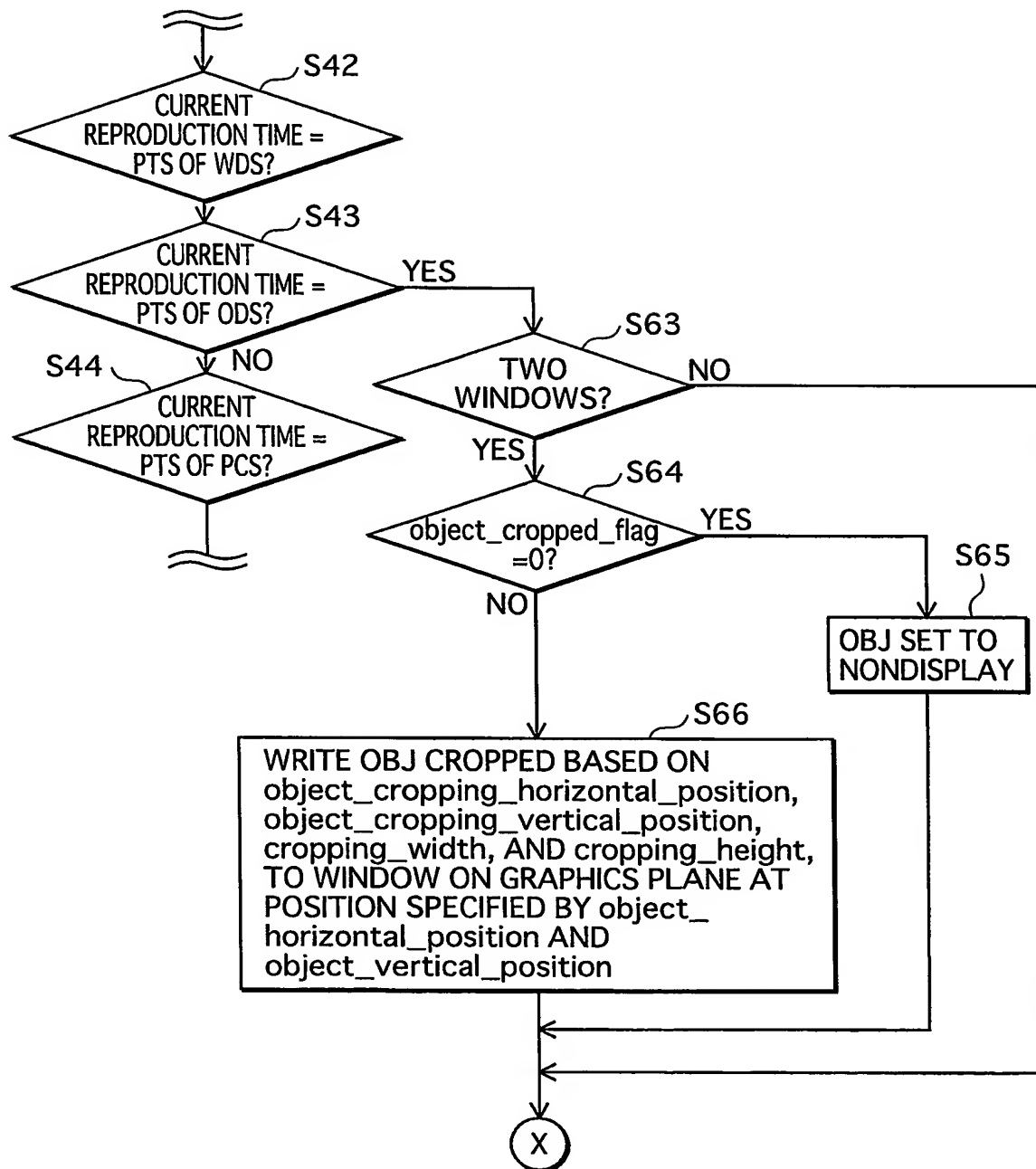




FIG. 41

